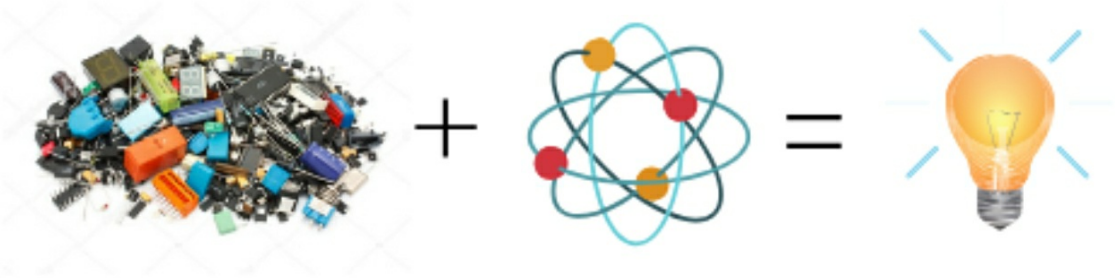
Hacking Physics Journal

VOL.1, NO 3 JULY 2020

Low-Cost Precision Voltage and Temperature Measurements

ERIC BOGATIN



The HackingPhysics Journal

Vol.1, no 3

July 2020

Low-Cost Precision Voltage and Temperature Measurements

Eric Bogatin

www.HackingPhysics.com

The HackingPhysics Journal,

A publication of Addie Rose Press

The HackingPhysics Journal

The HackingPhysics Journal is a periodic publication from the Hacking Physics Lab, published by Addie Rose Press.

Each issue focuses on a specific topic related to the combination of low-cost electronics and physics to empower the reader to dive into more detail exploring the world around us.

Every issue includes all the details about the hardware and software tools enough of the principles to use them effectively, examples of science experiments anyone can do with these tools and some indications of experimenters to explore further.

Finally, each report that makes up each issue is meant as the missing manual for these sorts of experiments.

Table of Contents

Chapter 1. Measure Microvolts for $10 in About Five Minutes

- 1.1 The ADS1115 module
- 1.2 The instant shield module for quick prototyping
- 1.3 Installing the ADS1115 library
- 1.4 Taking first data

Chapter 2. Characterizing the ADS1115

- 2.1 Noise analysis
- 2.2 Measuring the ADC with a scope application
- 2.3 Noise reduction with averages
- 2.4 Noise on single-ended measurements
- 2.5 Noise measurements of an AD584 voltage reference
- 2.6 Mutual-aggression noise
- 2.7 Calculating the first harmonic component
- 2.8 Summary of the noise specs of the ADS1115

Chapter 3. Voltage Stability of Voltage Sources

- 3.1 The AD584 voltage reference
- 3.2 Experiment: Two independent AD584 references
- 3.3 Experiment: An external voltage source from a function generator
- 3.4 Experiment: An AA battery
- 3.5 Experiment: 5 V USB hub
- 3.6 Experiment: 5 V rail from the LDO
- 3.7 Experiment: A resistor pot and the 5 V LDO as a reference offset voltage
- 3.8 Experiment: Resistor voltage divider
- 3.9 Experiment: voltage across a 2.2 V Zener diode with 2 mA forward current
- 3.10 Experiment: Measuring the forward voltage drop of a silicon diode, 1N7000
- 3.11 Experiment: forward voltage drop of an ultra-bright white LED

3.12 A hint at temperature measurements
3.13 Summary of the lessons learned so far
Chapter 4. The Complete Sketch for Data Acquisition and Analysis
4.1 Setting up for typical measurements
4.2 The experimental measurement system
4.3 Overview of the sketch
4.4 The main section
4.5 Calculate rms values
4.6 Measure with the ADS1115
4.7 Printing to excel
4.8 Taking data fast like a scope
Chapter 5. Building a Simple Four Channel Temperature Measurement System
5.1 The TMP36 temperature sensor
5.2 Configuring the TMP36 sensor
5.3 First temperature measurements
5.4 Using a USB cable for remote sensing
Chapter 6. Voltage Noise in the TMP36
6.1 Measuring the rf noise in my lab
6.2 Measured noise on the TMP36 sensor
6.3 Reducing the rf pick up noise in the sensor
6.4 The impact of reducing rf pick up
Chapter 7. Calibrating the temperature
7.1 Accuracy vs precision
7.2 Calibrating the TMP36 with two temperatures
Chapter 8. Stable temperature environments
8.1 Temperature stability of an ice bath
8.2 Ice bags in an insulating box
8.3 Temperature stability in ambient air
8.4 Cooling of coffee cups

Chapter 9. Conclusions
Chapter 10. Your Next Steps

Chapter 1. Measure Microvolts for $10 in About Five Minutes

For less than $10 you can build an incredibly powerful data acquisition system capable of measuring a microvolt change with a 1 second averaging time from four different channels.

You can build this system in about 5 minutes and measure precision voltages up to 5 V with a noise floor as small as 1 uV with an absolute accuracy of 1 mV.

When measuring a temperature sensor, this can translate into temperature changes smaller than 1 milli degC.

With open source code and the sketches in this issue of the HackingPhysics Journal, you can immediately take measurements and automatically post them to an excel spreadsheet, plot them and analyze them in real time, all for under $10.

This issue of the HackingPhysics Journal, vol 1, no.3 July 2020, covers how to:

- ✓ *Set up and use the ADS1115 16-bit ADC with 16x programmable gain amplitude*
- ✓ *Measure an absolute voltage accuracy to better than 1 mV*
- ✓ *Measure the noise floor to be lower than 10 uV rms*
- ✓ *Automatically collect data, print it to excel, plot it, save it*
- ✓ *Measure a TMP36 temperature sensor to see temperature changes smaller than 0.05 degF.*
- ✓ *For an entire system cost less than $10*
- ✓ *And all the sketches to perform these measurements are included in this issue.*

1.1 The ADS1115 module

In the Jan 2020 issue of the HackingPhysics Journal, we introduced the

ADS1115 module, a 16-bit analog to digital converter (ADC) module with four single-ended or two differential channels.

In front of the ADC is a programmable gain amplifier (PGA) with the highest gain setting of 16x. It communicates over the I2C bus so can talk to almost any Arduino. The default sample rate is 128 Samples per second (SPS).

On the ±0.256 V input voltage scale which uses a gain of 16x in the programmable gain amplifier, we can record voltage steps as low as 7.8 uV as the least significant bit (LSB) and an rms noise less than about 7.8 uV/2 = 3.9 uV taking data at 128 Samples per second, SPS. It decreases with averaging from here.

The ADS1115 is a TI chip that can be purchased on a small module with 100-mil center pins which can be inserted into a solderless breadboard. You can download the datasheet here. The module can be purchased here for less than $2.

A complete module used in all the experiments in this issue, with the I/O pins ready to insert into a solderless breadboard, is shown in Figure 1.1.

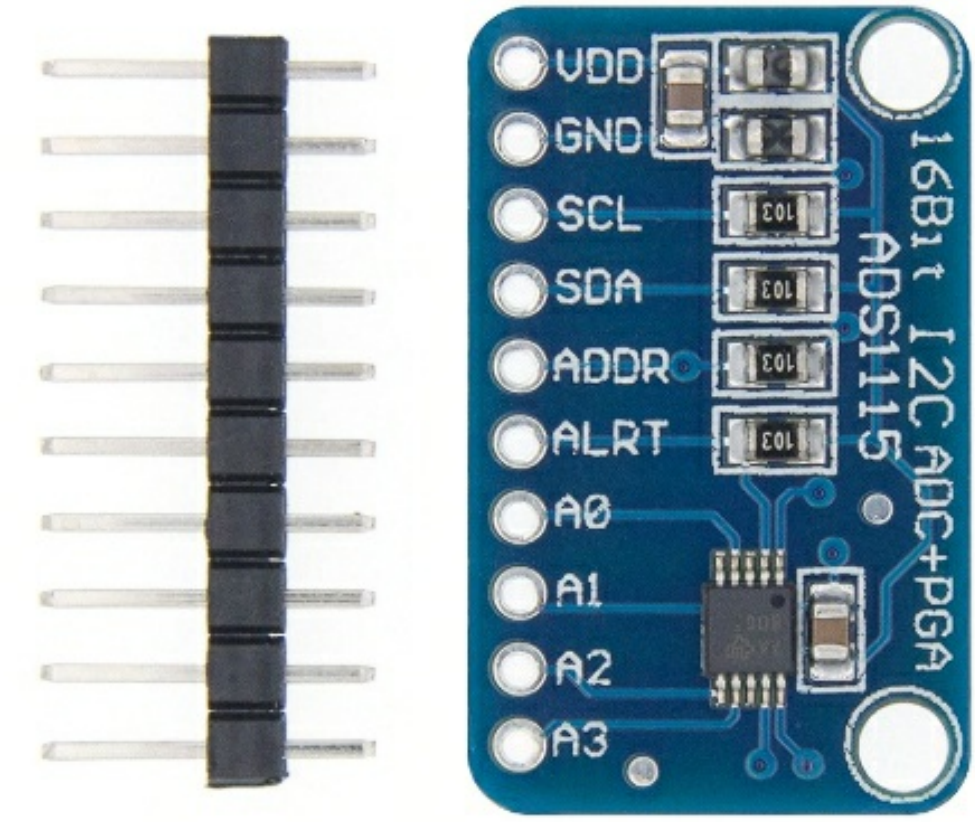


Figure 1.1 The ADS1115 module used in all of these experiments.

This module has four analog inputs which can be configured as two differential inputs or four single-ended inputs using an internal multiplexer. The block diagram is illustrated in Figure 1.2.

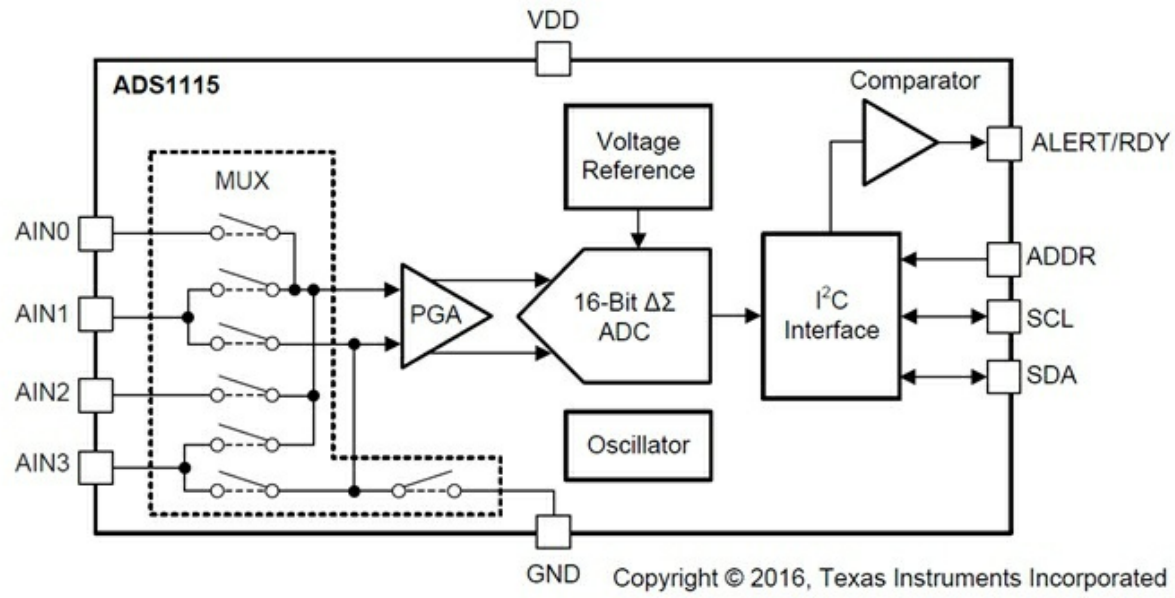


Figure 22. ADS1115 Block Diagram

Figure 1.2 Block diagram of the inside of the ADS1115, from the datasheet.

When the inputs are selected as differential, the adjacent two pins, A0 and A1 or A2 and A3, are connected into the programmable gain amplifier (PGA). The difference voltage between these two inputs is amplified by the (PGA) and its output measured by the 16-bit ADC.

For a single-ended input measurement, the + input to the PGA is connected to the measured channel, while the – input to the PGA is connected to the local ground connection in the package. In a single-ended measurement, it is the voltage difference between the input of the channel and the local ground that is amplified and measured.

The PGA can be set for six different gains, 2/3, 1x, 2x, 4x, 8x and 16x. These different gains result in a full-scale voltage input range and resulting least significant bit (LSB) level shown in Figure 1.3.

Table 3. Full-Scale Range and Corresponding LSB Size

FSR	LSB SIZE
±6.144 V(1)	187.5 μV
±4.096 V(1)	125 μV
±2.048 V	62.5 μV
±1.024 V	31.25 μV
±0.512 V	15.625 μV
±0.256 V	7.8125 μV

Figure 1.3 Full scale voltage range and the resulting least significant bit level from the datasheet.

There is an internal bandgap voltage reference which means the ADS1115 requires no calibration. It's specified absolute accuracy is better than 0.1%.

1.2 The instant shield module for quick prototyping

Unfortunately, the pin out of the module does not make it easy to plug directly into the Arduino pins. This means we have to use an external solderless breadboard to plug the module into and then use jumper wires to the Arduino pins.

There are only four connections needed from the Arduino to the ADS1115: 5 V, gnd, SCL, SDA.

- *+ 5 V: the 5 V rail from the Arduino board*
- *Gnd: the ground connection to the Arduino board*
- *SCL: the serial clock line of the I2C bus which connects to either the SCL pin above pin 13 or alternatively, to the analog A5 pin*
- *SDA: the serial data line of the I2C bus which connects either to the SDA pin above pin 13 or alternatively, to the analog A4 pin.*

On the ADS1115 module, the address pin should go to the gnd pin as the default address. The alert pin should be left not connected. It will not be used when sampling at 128 SPS.

The other four pins left are the A0 to A3 analog input pins.

One simple way of making all of these connections is using a mini, instant shield that can plug directly into the Arduino header sockets. An example of an instant shield available from here for only $2 is shown in Figure 1.4.

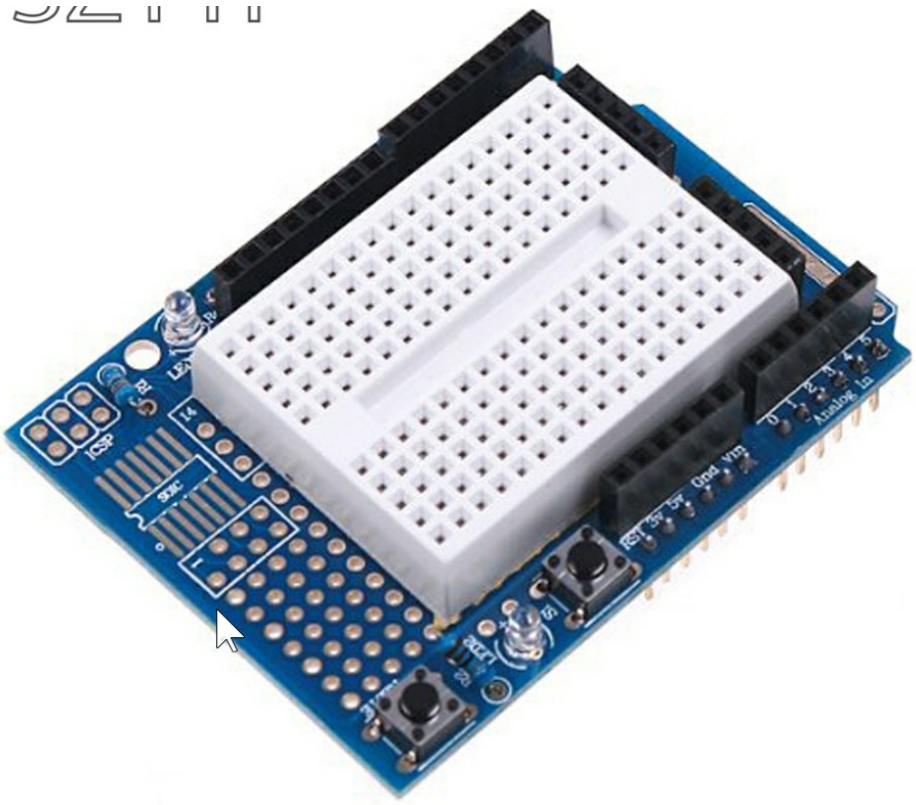


Figure 1.4 Example of the instant shield which can attach directly into an Arduino.

This is the simplest way of connecting an ADS1115 into an Arduino. Wire the module into the solderless breadboard and connect the four Arduino pins to the header pins. Then when the module is plugged into an Arduino Uno, the four connections to the Arduino are automatically connected. An example of an instant shield configured for an ADS1115 is shown in Figure 1.5.

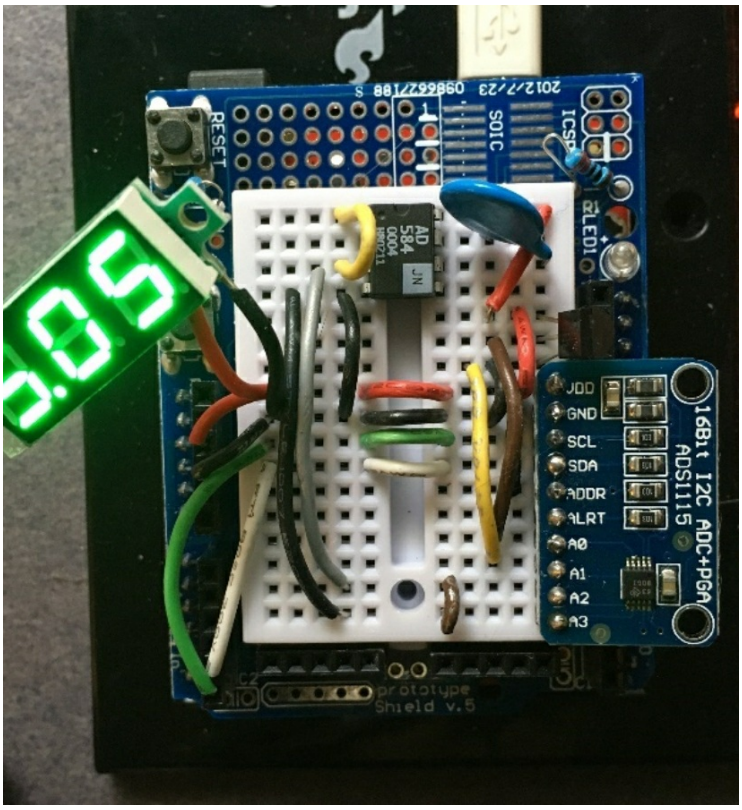


Figure 1.5 Closeup of a mini instant shield with the ADS1115 wired in and an AD584 and TMP36 sensor all in one module. The shield is inserted into a Sparkfun Redboard.

The advantage of using this instant shield is that this module is transportable. It can be unplugged and moved to other Arduinos and even connected remotely to an Arduino by using just the four wire connections needed.

Depending on the Arduino Uno you use, its price can range from $4 to $20. The total cost of this system used for the basic measurements is:

ADS1115	$2
Mini instant-shield	$2
Arduino Uno	$4
Total	= $8

1.3 Installing the ADS1115 library

There is a simple to use library available for the ADS1115 available from Adafruit. The library manager can be used to install it. This is the simplest to

use and perfectly adequate library to drive the ADS1115. Installing this library allows using the Adafruit_ADS1x15.h library.

However, there are a few additional high-level commands available in another library that I use throughout these experiments. It uses the same library name as the Adafruit basic library.

This new library allows setting the sample rate and reading the mV/ADU bit level after the gain has been set.

You can download the zip version of this library here: https://github.com/soligen2010/Adafruit_ADS1X15

The zip file is labeled: Adafruit_ADS1X15-master.

Once you download the .zip file, it is very simple to install it.

However, since this library uses the same library name as the Adafruit driver, you first have to uninstall the Adafruit default library. Just drag its folder out of the Arduino library folder.

The Arduino uses a default location for its libraries. It is usually in the Documents folder, under Arduino/libraries. To install my preferred library, just move this zip file to the library directory.

Next, in the Arduino IDE, select sketches/Include Library and select add zip library, as shown in Figure 1.6.

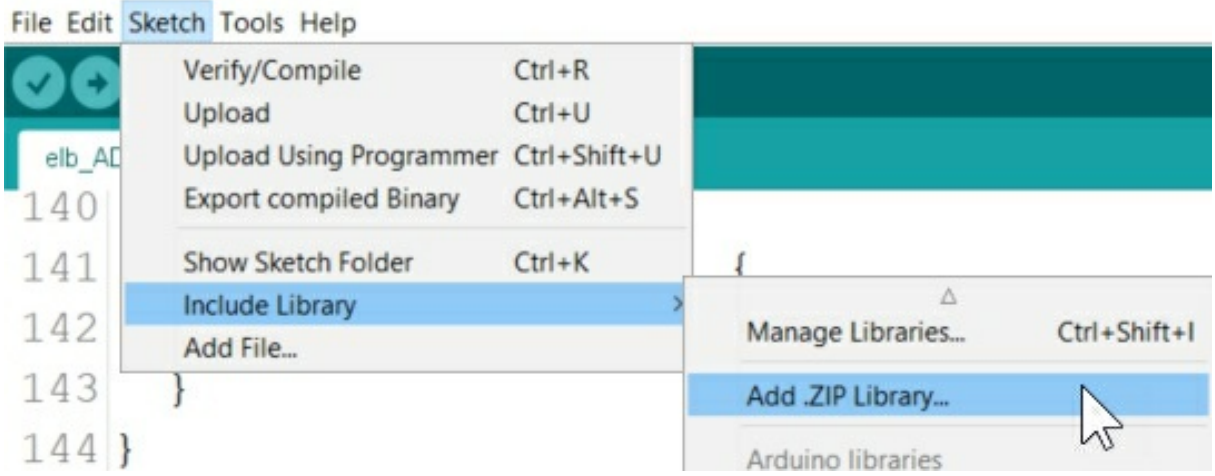


Figure 1.6. To install the .zip library, select the add .zip library option.

Browse to the documents/Arduino/libraries and select the Adafruit_ADS1X15-master zip file.

That's it. It is now installed.

The improved library is labeled as Adafruit_ADS1X15-master.zip. However, the name of the library inside this folder is identical to the Adafruit default library.

1.4 Taking first data

To take data at the highest rate of 860 SPS requires connecting the ALERT pin of the ADS1115 to one of the digital I/O pins and using interrupts to monitor this pin and read the measurement when the ADS1115 is ready.

However, there will be more digitizing noise using a higher data rate than 128 SPS. When we plan to take data every second or so, there is no additional value in reading from the ADS1115 at 860 SPS compared to 128 SPS, so this higher sample rate is not needed.

Instead, we will plan on taking data at the default rate of 128 SPS and doing external averaging in the sketch to reduce the noise.

The PGA gain setting depends on the highest voltage level we expect to see. When measuring a 2.5 V reference, for example, the most sensitive scale that fits in this voltage range is the ±4.096 V scale which is a gain setting of 1x.

When measuring a TMP36 temperature sensor, the typical voltage is less than 0.8 V, so the most sensitive scale to use is the ±1.024 V full scale which is a gain setting of 4x.

As a first measurement, I connected in a TMP36 temperature sensor into channel 0 of the ADS1115. I set the gain to 4x on the PGA.

Using this as the basis, we can create a minimalist sketch to take data as a single-ended measurement from channel A0 and plot these measurements. The sketch is here:

```
// www.HackingPhysics.com data acquisition
// Eric Bogatin
// minimalist ADS1115 sketch
```

```
#include <Wire.h>
#include <Adafruit_ADS1015.h>
Adafruit_ADS1115 ads;

void setup() {
   Serial.begin(2000000);
   ads.begin();
   ads.setSPS(ADS1115_DR_128SPS);
}

void loop() {
   ads.setGain(GAIN_FOUR);
   Serial.println(ads.readADC_SingleEnded(0)*ads.voltsPerBit() * 1000.0F);
   delay(100);
}
```

One advantage of this particular ADS1115 library is that it gives us access to the new command, ads.voltsPerBit() which will read the conversion factor of volts per bit based on the gain setting. This simplifies the conversions.

Throughout this sketch and all future ones, I use mV as the sacle for a voltage value.

By running this sketch, we can plot on the serial plotter the ADU values, then convert into mV values, with an artificial delay of 100 msec between points. An example of a typical measurement is shown in Figure 1.7.

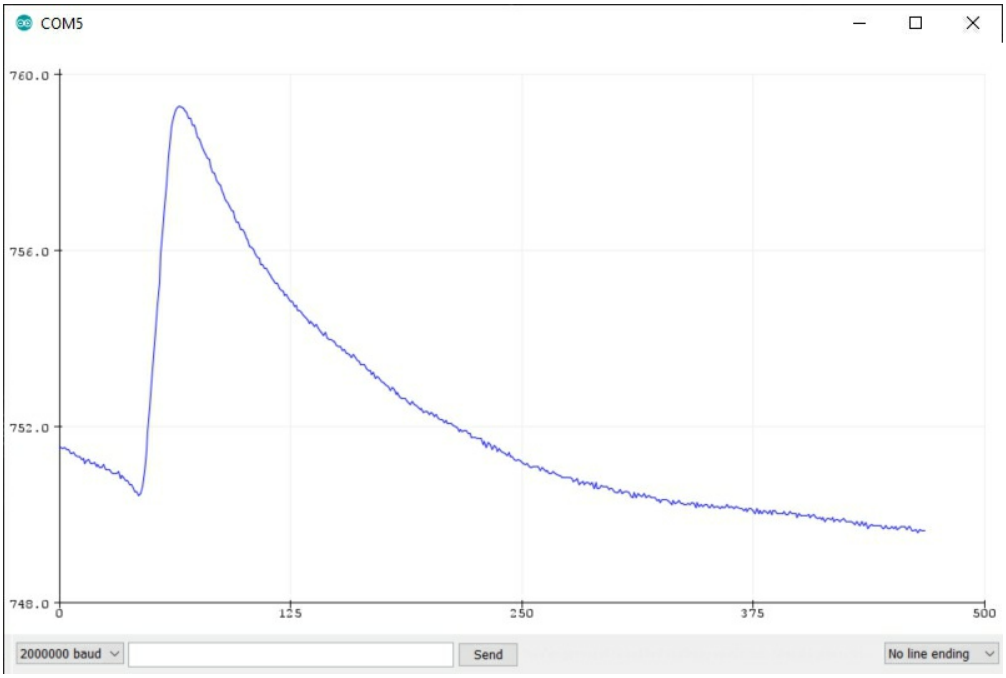


Figure 1.7 Measured voltage out of the ADS1115 when measuring the voltage output of a TMP36

temperature sensor. The vertical scale is 4 mV/div. The sensitivity of the sensor is 10 mV/degC. The vertical scale corresponds to 0.4 degC/div.

For this sensor, the sensitivity is 10 mV/degC. This is 100 degC/V or 0.1 degC/mV. The vertical scale on the plotter is 4 mV/div. This is 0.4 degC per division.

After touching the TMP36, the voltage shot up from about 749 mV to 759 mV. This is a change of about 10 mV which is 1 degC, temperature change.

This is the motivation of using this ADS1115 to measure the temperature response of the TMPR36. It has an incredible sensitivity with little effort.

On the scale of ±1.024 V, the LSB level is 31 uV = 0.031 mV. This corresponds to a LSB level for a temperature measurement of 0.1 degC/mV x 0.031 mV = 0.003 degC as the baseline LSB sensitivity. This is a really small value. We expect the noise or fluctuations in the measurements will be much larger than this.

This first measurement also indicates that the thermal response time for the sensor is about 2-3 seconds heating up and 10 seconds cooling down in air. This suggests that an averaging time of about 1 second should be perfectly adequate in most temperature measurements.

In this first measurement, I added a 100 msec delay slowing the rate at which measurements were plotted. With 500 points on the horizontal scale, this is 50 sec full scale or 12.5 sec per division.

If we want to end up with measurements at a 1 second interval, we will take data at the rate of 128 SPS, as fast as possible, and average them over this time interval to reduce the noise.

As a general principle, we never want to leave measurements on the table, but use every last drop of data we can.

Chapter 2. Characterizing the ADS1115

The datasheet states that the noise from the ADS1115 should be less than 1 LSB when measuring at a sample rate of 128 SPS or less.

It is always valuable to measure whatever features we can think of to verify the specifications or clarify them. This is part of the process of reverse engineering.

We can get a measure of the intrinsic noise floor of the ADS1115, some techniques that increase the noise and the absolute accuracy of the ASD1115 with a few simple experiments.

2.1 Noise analysis

There are generally two types of noise to consider in any device that has multiple input channels: self-aggression noise and mutual-aggression noise.

The self-aggression noise is the noise from the channel itself, on its own measurements. This is either the amplifier noise or the digitizing noise. Mutual-aggression noise is the cross talk between the voltage on one channel that may leak into another channel.

Generally, if this noise is random, Gaussian, or white noise, the rms value of the noise (self-aggression noise) is a good figure of merit to describe the noise.

The rms value is defined as:

$$rms = \sqrt{\frac{1}{n}\sum_{i} V^2(i)}$$

Where

n = number of points recorded

V(i) = the voltage measured on consecutive measurements

This means that if we have an offset or average value to the voltage

measurements, this will be included in the rms value. To avoid this problem, so that we only see the noise, we really want to calculate what is sometimes referred to as the AC-rms value, or the standard deviation.

This is just the square root of the sum of the squares of the voltage with the average value subtracted off. The standard deviation, or the 1 sigma value is:

$$sigma = \sqrt{\frac{1}{n}\sum_i \left(V(i) - \langle V(i) \rangle\right)^2}$$

To calculate the standard deviation, we have to calculate the average of the dataset, subtract this from each value and square the difference, add them up and divide by the number of points in the distribution and then take the square root.

In order to calculate the average value, I need to collect the entire dataset. Then calculate the average and subtract this from each elements.

In the sketch I wrote, I used an array to store the data set. This allows me to do some statistics on the dataset after I have recorded the entire set.

Each point in the array is itself the average of npts_scope_ave individual measurements, taken as quickly as possible, using a sample rate of 128 SPS.

The actual raw sample rate from the ADS1115 I get with this sketch, is 108 SPS going at the fastest rate I can take data. This is still perfectly adequate.

2.2 Measuring the ADC with a scope application

As a starting place, I wrote a sketch which emulates a scope. I take 300 consecutive measurements and load them into an array, with the minimum delay.

Unfortunately, using an Arduino Uno with an Atmega 328 uC, the maximum size array I can use is about 300 points. Since each measurement is 16-bit resolution, each element has to be either a long integer or float. If we need a larger array, we have to switch to a different Arduino, like the SAMD21. But, for all the experiments in this issue. The Uno is perfectly adequate.

Once I have the array of 300 consecutive measurements filled, I can print the

measurements into excel just like a scope display and do calculations on them. When using the PLX-DAQ macro in excel, described in HackingPhysics Journal vol 1, no 1, Jan 2020, the fastest we can write to excel is about 6 points per second.

This means we need to add a delay when writing the scope measurements into excel.

The next question is what voltage source to measure when looking at the intrinsic noise level. The absolute lowest noise will be when the inputs to a different channel are shorted together and connected to external ground.

The equivalent input impedance model for the differential channels is shown in Figure 2.1.

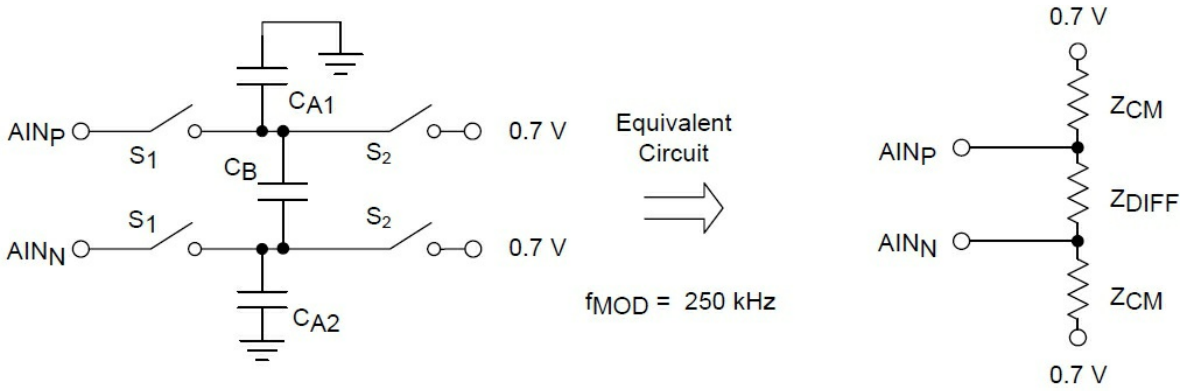


Figure 2.1 Equivalent input impedance model for the differential circuits. The differential and common impedances are 3 M Ohms or larger on all scales, taken from the datasheet.

When we short together the inputs of AINp and AINn, we short the internal 3 Mohm differential resistance and the inputs are biased to 0.7 V of common voltage. We should measure a slight DC difference in voltage between the AINp and AINn depending on the balance of the internal circuit. But all we want is the noise in this difference.

Since this voltage difference is so small, we can use the most sensitive scale with a gain of 16x.

This action of taking data quickly and sending it to excel was written as a function. This is a modular piece of code which can be written and placed after the void loop() section, which is itself, a function.

In the code I wrote, I used tabs to place the functions. This makes the sketch

neat and easy to manage.

The function to take a buffer of measurements, and print the results to excel is here:

```
void func_scope() {

    if (iFlag_XL == 1) {
        // if we print to excel, set it up
        Serial.println("CLEARRANGE,A,1,C,9001");
        Serial.println("LABEL, index, Time(sec), mV");
  }

    // initialize array
    for (int i = 1; i <= npts_scope_buffer; i++) {
        arrVscope_ADU[i] = 0.0;
  }
    ads.setGain(GAIN_SIXTEEN);

    //begin quickly filling int array with averaged points
    iTime_start_usec = micros();
    for (int i = 1; i <= npts_scope_buffer; i++) {
        for (int j = 1; j <= npts_scope_ave; j++) {
            //arrVscope_ADU[i] = arrVscope_ADU[i] + ads.readADC_SingleEnded(pinADS1115);
            //arrVscope_ADU[i] = arrVscope_ADU[i] + ads.readADC_Differential_0_1();
            arrVscope_ADU[i] = arrVscope_ADU[i] + ads.readADC_Differential_2_3();
    }
        arrVscope_ADU[i] = arrVscope_ADU[i] / (npts_scope_ave * 1.0);
  }
    Time_X_usec = (micros() - iTime_start_usec) * 1.0;
    Time_X_usec = Time_X_usec / (npts_scope_ave * npts_scope_buffer * 1.0);

    // begin printing out formated array contents
    for (int i = 1; i <= npts_scope_buffer; i++) {
        V_meas_ADU = arrVscope_ADU[i];
        V_meas_mV = V_meas_ADU * ads.voltsPerBit() * 1000.0F;;
        V_meas_degC = V_meas_mV / 10.0 - 50.0;
        V_meas_degF = V_meas_degC * 1.8 + 32.0;
        Time_point_msec = i * Time_X_usec * 1.0e-3;
        Time_point_sec = i * Time_X_usec * 1.0e-6;
        if (iFlag_XL == 1) {
            // if printing to excel, format the data
            Serial.print("DATA,");
            Serial.print(i); Serial.print(", ");
            Serial.print(Time_point_sec, 5); Serial.print(", ");
            //Serial.println(Temp_degF, 3);
            Serial.println(V_meas_mV, 4);
            delay(delay_XL_msec);
    }
        else {
            //if not printing to excel, format for serial monitor
            Serial.print(i); Serial.print(", ");
            //Serial.print(Time_X_usec); Serial.print(", ");
            Serial.print(Time_point_msec, 3); Serial.print(", ");
            //Serial.print(100); Serial.print(", ");
            //Serial.print(V_meas_ADU, 3); Serial.print(", ");
            Serial.println(V_meas_mV, 4);
            //Serial.println(V_meas_degC);
            //Serial.println(V_meas_degF);
    }
  }
    // done exporting the array
    if (iFlag_XL == 1) {
        Serial.println("SAVEWORKBOOK");
        while (1 != 2) {
            Serial.println("BEEP");
            delay(1000);
    }
  }
    delay(2000000); // wait to view screen
}
```

To run this code, you can literally copy it from this ebook and paste it into a blank sketch. You may have to clean out and page header information that also got copied over. But otherwise, it will just run. See this blog post as an

example.

The measured voltages with the inputs to channels A2 and A3 shorted together and connected to ground, as plotted in excel, are shown in Figure 2.2.

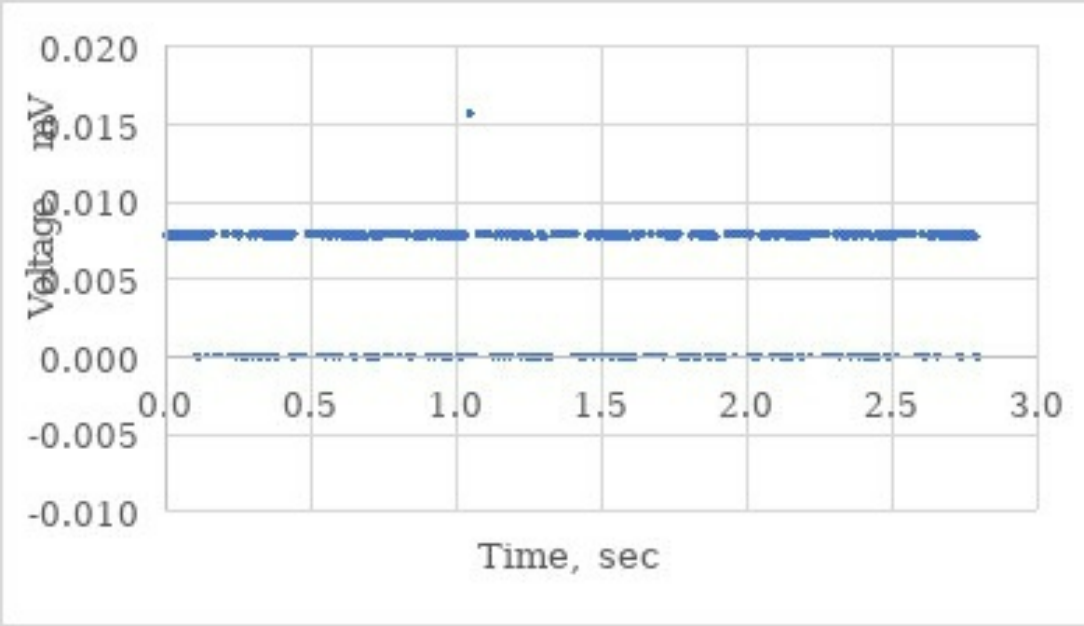


Figure 2.2 Measured differential voltage at 108 SPS with inputs shorted of the scale of 16x. The scale for the plot is 5 uV/div.

On this scale of ±0.256 V full scale, the LSB value is 0.0078 mV. A glance at the measured voltage over this 2.8 sec time interval, the values measured are all within ±1 LSB of each other.

The calculated rms value for this data set is 0.0038 mV. This is about ½ of the LSB value. This is a good measure of the expected digitizing noise we might expect to see.

2.3 Noise reduction with averages

With the sample rate set at 128 SPS, which in practice, is 108 SPS, we get a typical rms noise level of about ½ LSB. If we use a faster sample rate, the noise will increase. If we used a slower sample rate, which means there would be more internal averaging, the noise would be less than ½ LSB and we would not see any noise. We would not be able to see a noise reducing

with averaging.

The sample rate of 108 SPS is the optimum sample rate. If we start with a noise rms value of ½ LSB, as we average more points together, we expect the resulting rms value will decrease. If the noise is white noise or Gaussian, the rms value will decrease with the square root of the number of averages.

The rms(n_averges) is:

$$rms(n) = \frac{rms(1)}{\sqrt{n}}$$

To test this, I measured n consecutive points taken at 108 SPS and averaged them together. Then I measured 300 consecutive averages and loaded them into an array. To each point in the array, I subtracted the group's average and calculate the rms value of the distribution.

I expected the rms value to decrease with the square root of the number of averages. Along with the rms value for each number of averages, I calculated the model based on starting with an rms value of ½ the LSB and decreasing with the square root of the number of averages.

The function in the sketch that performs these calculations and prints the number of averages and the rms value to the serial monitor is here:

```
//calc the rms noise with npts_ave

void func_calcRMS() {
    ads.setGain(GAIN_SIXTEEN);
    arr_mV_per_ADU[0] = ads.voltsPerBit() * 1000.0F;

    for (npts_scope_ave = 1; npts_scope_ave <= 500; npts_scope_ave++) {
        ADS_meas_ADU_ave = 0.0;
        ADS_meas_ADU_sigma = 0.0;

        for (int j = 1; j <= npts_scope_buffer; j++) {
            ADS_meas_ADU = 0.0;
            iCounter1 = 0;
            for (int i = 1; i <= npts_scope_ave; i++) {
                //ADS_meas_ADU = ads.readADC_SingleEnded(pinScope) * 1.0 + ADS_meas_ADU;
                ADS_meas_ADU = ads.readADC_Differential_2_3() * 1.0 + ADS_meas_ADU;
                iCounter1++;
     }
            arrVscope_ADU[j] = ADS_meas_ADU / (iCounter1 * 1.0);
            ADS_meas_ADU_ave = ADS_meas_ADU_ave + arrVscope_ADU[j];
    }
        ADS_meas_ADU_ave = ADS_meas_ADU_ave / (npts_scope_buffer * 1.0);
        for (int i = 1; i <= npts_scope_buffer; i++) {
            ADS_meas_ADU_sigma = ADS_meas_ADU_sigma + pow((arrVscope_ADU[i] - ADS_meas_ADU_ave), 2);
    }
        ADS_meas_ADU_sigma = sqrt(ADS_meas_ADU_sigma / (npts_scope_buffer * 1.0));
        Serial.print(npts_scope_ave); Serial.print(",  ");
        Serial.print((ADS_meas_ADU_sigma * arr_mV_per_ADU[0]), 5); Serial.print(",  ");
        Serial.print((ADS_meas_ADU_ave * arr_mV_per_ADU[0]), 5); Serial.print(",  ");
        Serial.println( );
  }
    delay(1e9);
}
```

Figure 2.3 is an example of the measured RMS noise value as the number of points externally averaged is increased. The straight-line model is for an rms value that drops with the square root of the number of points averaged, starting with an rms value with 1 average as ½ LSB level. The input measured is the differential voltage between the channel 2 and 3 inputs, shorted together and connected to ground on a scale of ±0.256 V full scale.

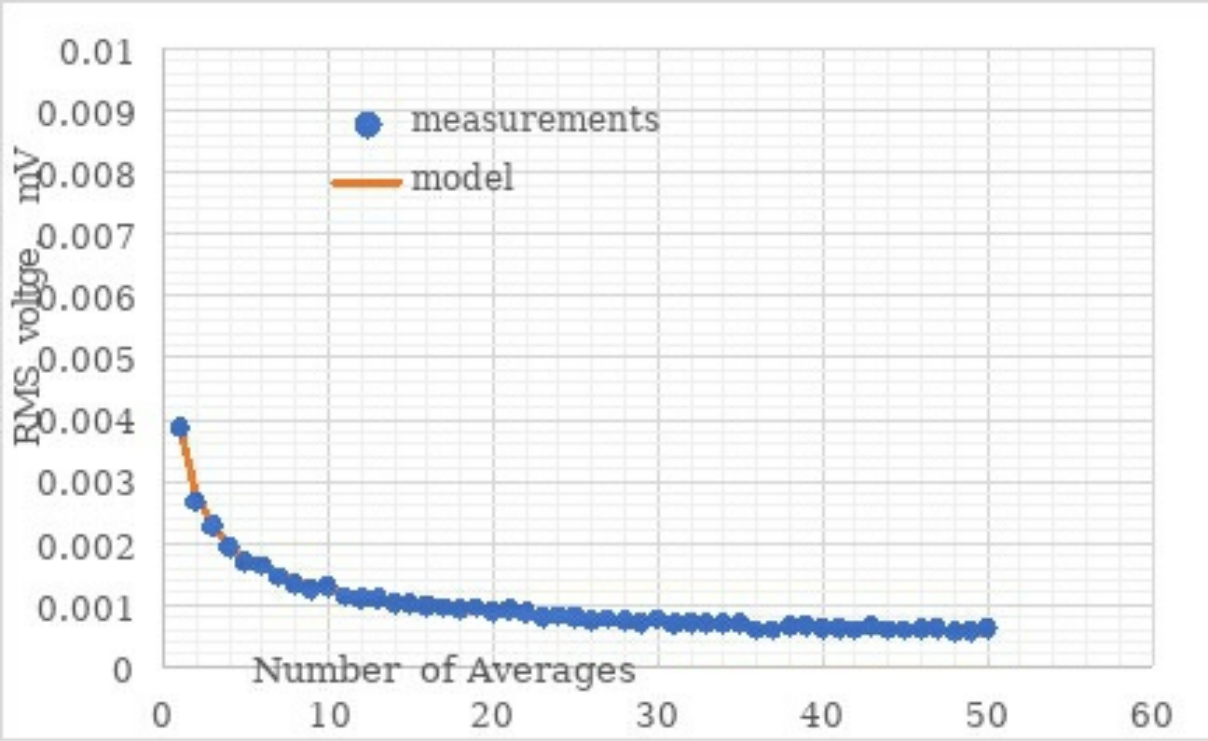


Figure 2.3 Measured rms value as the number of averages increased, compared with the simple Gaussian noise model using a starting rms value of ½ LSB.

The rms value behaves exactly as expected. The rms value starts at ½ the LSB value and decreases with the square root of the number of averages. This suggests the noise is white noise and is due to digitizing noise.

The change in the average value with error bars corresponding to ±1 sigma is shown in Figure 2.4.

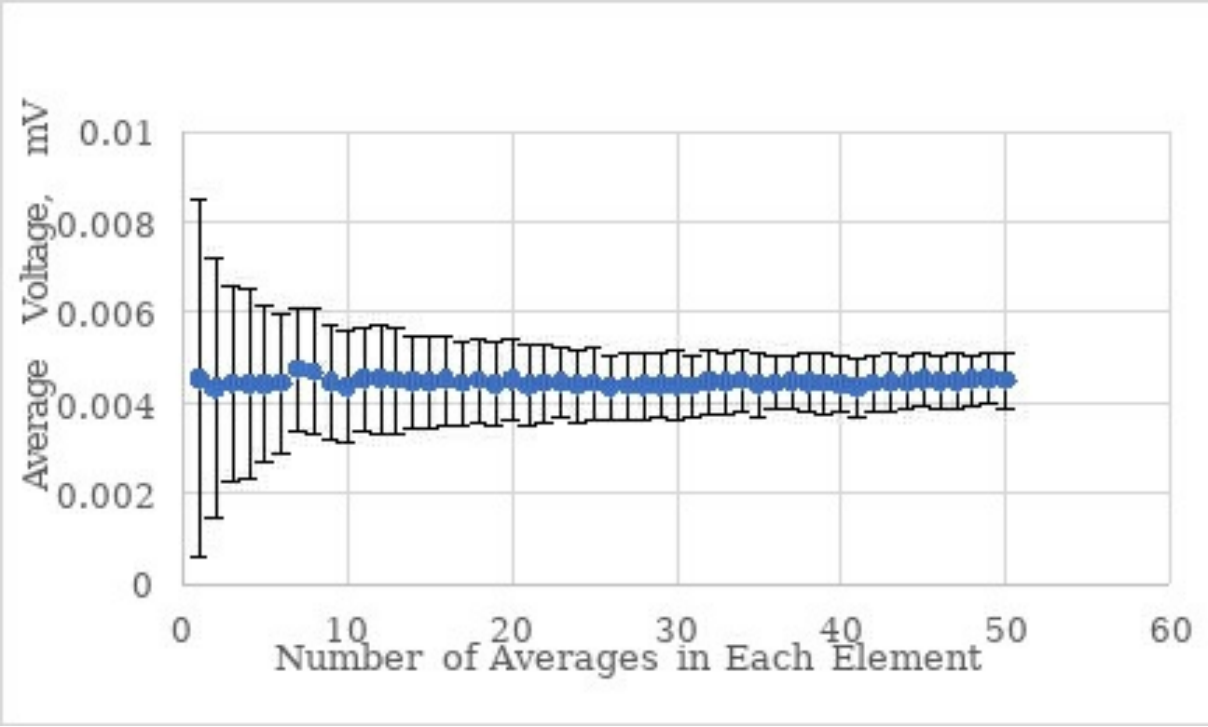


Figure 2.4 The measured average value with error bars corresponding to the 1 sigma rms noise for each measurement. The DC offset of the PGA is 4.5 uV.

Exactly the same noise is measured whether the inputs that are shorted together or also externally grounded, floating, or tied to 2.5 V DC. In all cases, there is an internal connection through 7 Mohm resistors to an internal voltage bias which provides a connection to a local voltage.

The important condition is the two inputs are connected together, and a differential measurement is used.

It is remarkable that with about 50 averages, or an average over ½ second, the rms noise level is 600 nV.

And this low a noise is measured with this very simple mini instant-shield plugged into the Arduino Uno powered with the 5 V rail of the USB hub with no special shielding or ground planes in vicinity.

It is a testament to how robust this ADS1115 module is and its potential for quick, simple, sensitive and accurate low-level voltage measurements.

2.4 Noise on single-ended measurements

In a differential measurement, the low side of the PGA connects to an external ground point on the board. Even if the two inputs are connected together, they are connected externally together. There is little voltage difference between them and what we measure as the difference voltage, is the intrinsic digitizing noise.

The rms noise at the fastest sampling rate of 108 SPS is ½ LSB or 7.8 uV/2 = 3.9 uV rms. This is specifically for a differential measurement.

In a single-ended measurement, the voltage of the low side of the PGA is connected to the local ground on the pad of the IC itself. This connects through a lead in the package to the ground on the board. This is shown in Figure 2.5.

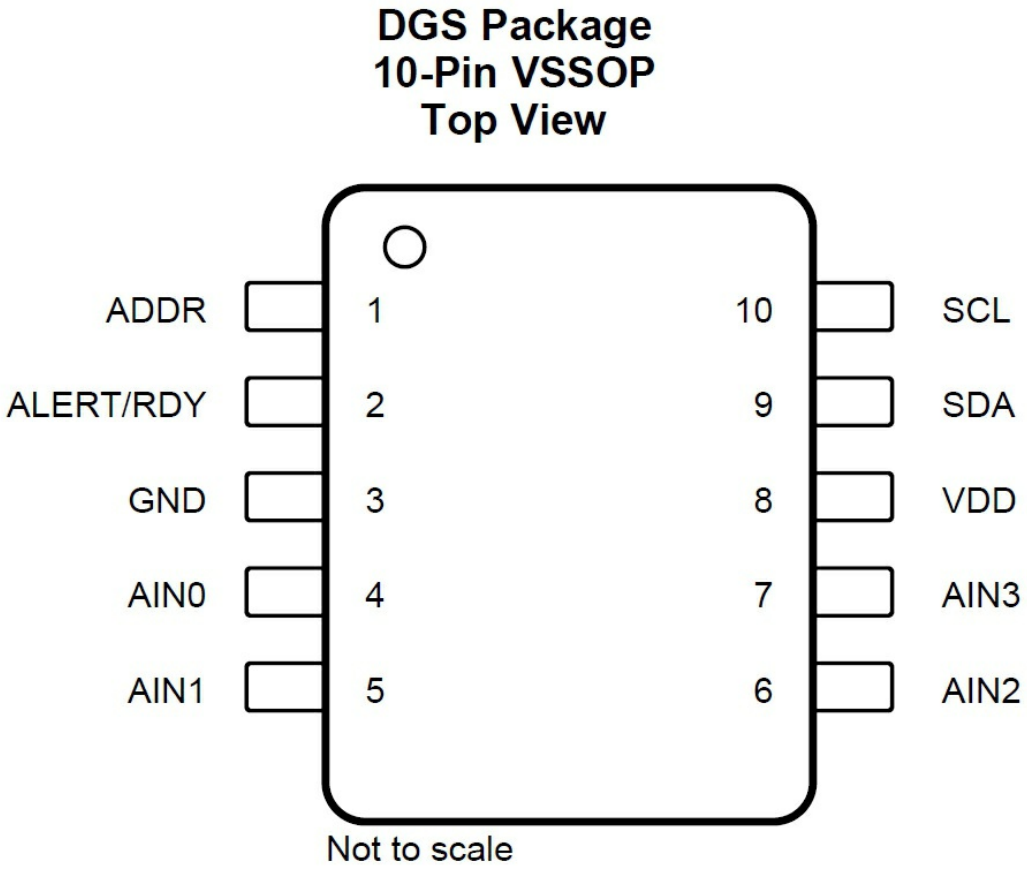


Figure 2.5 The pinout of the IC package for the ADS1115 chip itself. There is only one ground pin. This means all the digital and analog return current passes through this lead creating ground bounce between the internal gnd and the external ground.

The *internal* ground connection on the die and the *external* ground

connection will not be at the same voltage. The voltage difference between the internal ground point on the low side of the PGA and the external ground connection on the solderless breadboard is due to the ground bounce voltage between them.

There is only one ground lead in the ADS1115 IC package. The digital return current and the power return current will all pass through this same common ground lead. This current will pass through the resistance and inductance of the ground lead in the package. This current will generate voltage noise between the on-die gnd pad and the on-board gnd pad. This noise will appear between the low side of the PGA and whatever else is connected to the high side of the PGA. This circuit is shown in Figure 2.6.

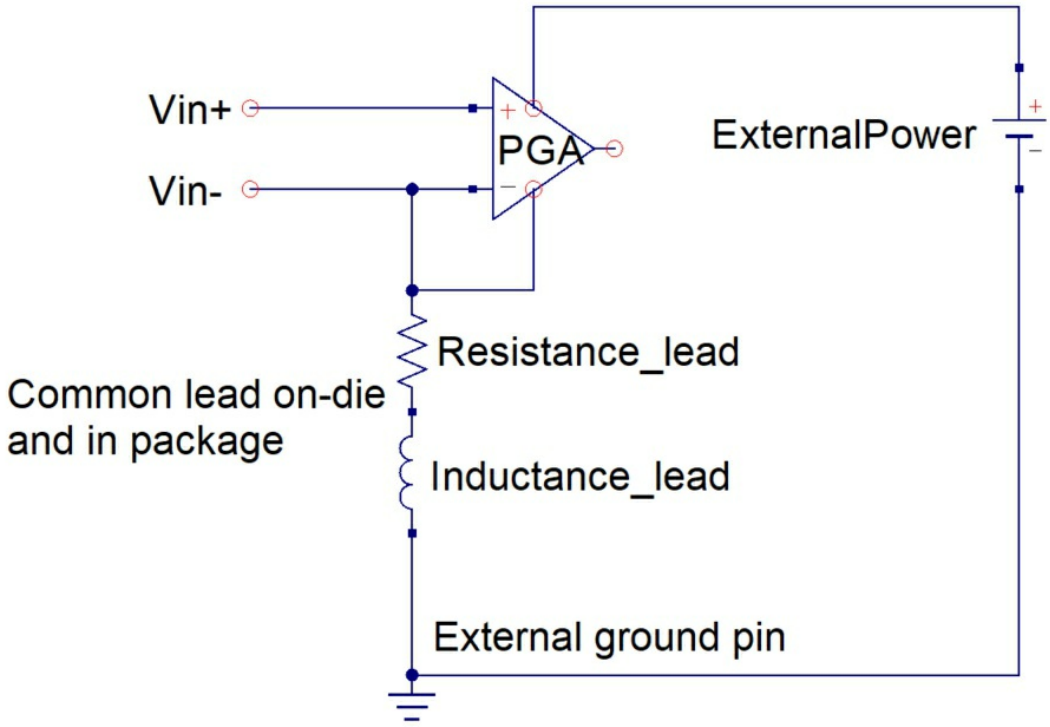


Figure 2.6 Equivalent circuit of the lead from the die to the external ground showing how it shares the path with the power return.

The power rail return current passing through the shared lead inductance and resistance will all contribute to a voltage difference between the internal gnd connection on the die. When the digital signals both from the core logic of the ADS1115 and the SCL and SDA lines switching will generating fluctuating or “switching” currents that will generate noise on this common

lead. We call this noise switching noise, and since it appears on the ground lead, we also refer to this as ground bounce.

When the low side of the PGA is connected to the internal ground point, the low side will see the ground bounce noise in addition to the signal from the high side input.

The noise on this ground lead due to the switching currents will appear as noise between the AINp and AINn pins of the differential amplifier. In fact, this is a simple way to measure the ground bounce noise in the package lead.

In higher performance devices, the analog ground, used as the input to the low side of the amplifier, is pulled out as a separate pin of the package from the digital return. This way the analog input does not see the ground bounce on the digital return pin. This is not the case in the ADS1115.

By connecting external ground to channel A3, and doing a single-ended measurement, we can measure the voltage noise between the internal ground pad, on the low side of the PGA and the external ground connected to the high side of the PGA. This measurement of the rms noise decreasing with the number of averages is shown in Figure 2.7. In this example, also plotted is a measure of the noise on a differential measurement.

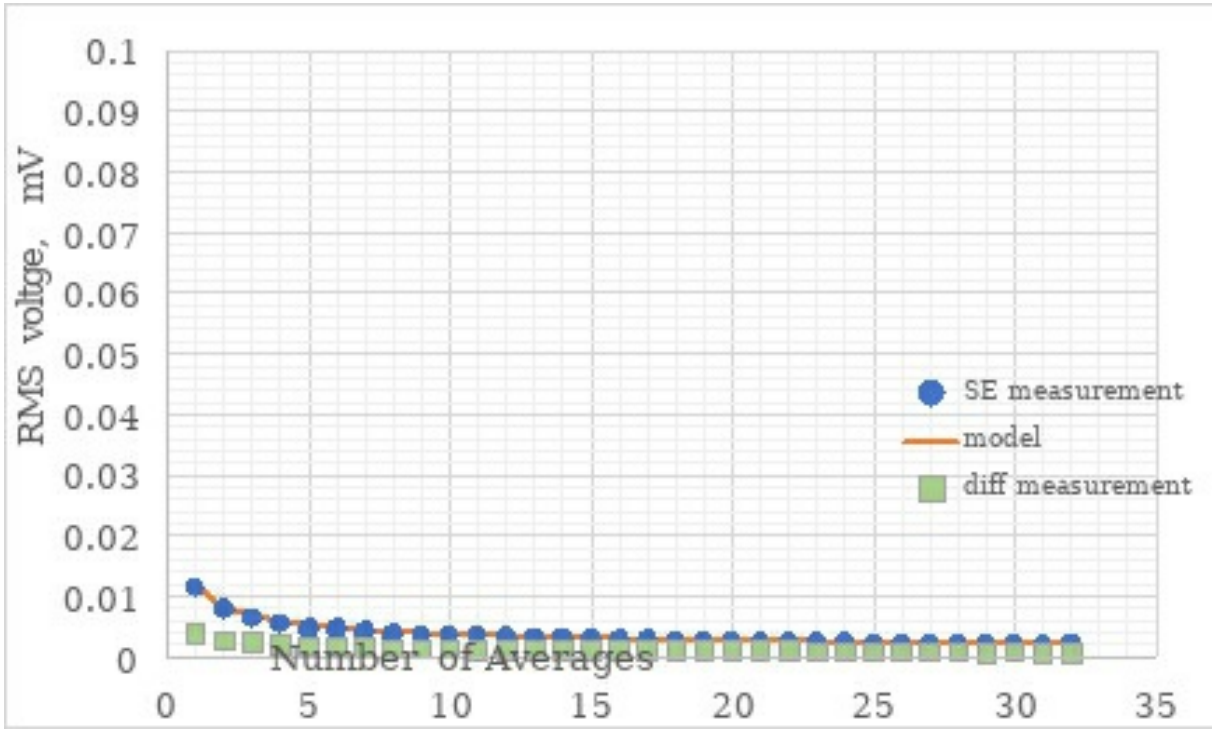


Figure 2.7 Measured rms noise with increasing number of averages for a single-ended measurement and a differential measurement of the external ground. This difference is a direct measure of the ground bounce noise on the common ground lead.

In this measurement, the rms noise level for the single-ended measurement is 11 uV for the fastest sampling rate of 108 SPS, compared with 3.9 uV rms noise for the differential measurement.

This difference is due to the ground bounce noise on the ground lead of the IC package created by all the switching return currents for the digital communications and the power return current.

The DC voltage measured on the single-ended measurement of the external ground is about -0.24 mV. This average is shown in Figure 2.8. It is due to the DC voltage drop from the average return current through the package lead.

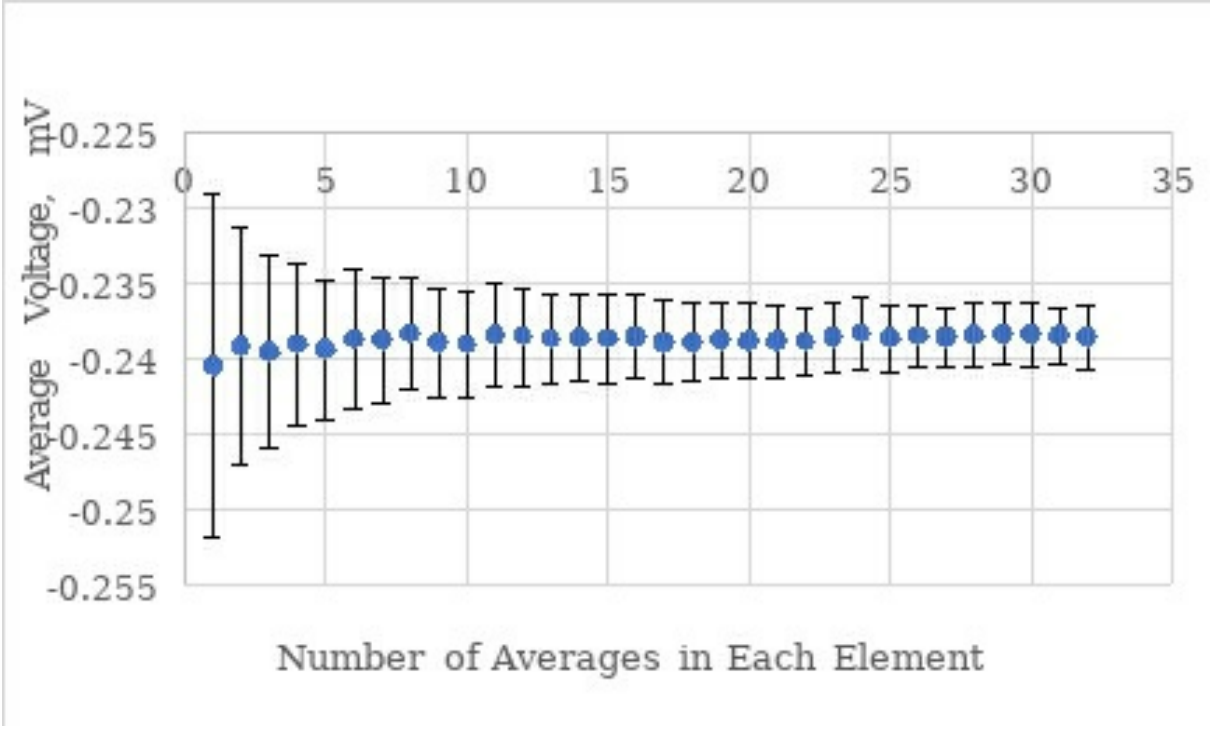


Figure 2.8 Averaged measured voltage on the single-ended measurement of external ground.

The negative voltage means the low side voltage is higher than the high-side voltage of the PGA. This is only 0.24 mV.

The rated DC power current for the ADS1115 is only 200 uA. If we assume there is a total of 0.5 mA due to the return current of the I2C bus, for

example, then this 0.24 mV DC offset corresponds to a lead resistance of R = 0.24 mV/0.5 mA = 0.48 Ohms.

The package lead resistance is probably on the order of 0.05 ohms, so this value of 0.48 ohms is probably due to on-die metallization resistance from the gnd pad to the location on the die for the input to the low side of the PGA, through which all the return current is flowing.

This observation suggests that for the lowest noise, we should use a differential measurement, even if the low side of the differential pair is connected to the local ground connection where the sensor is, rather than use a single-ended measurement.

If the 11 uV of rms noise, at the highest sampling rate is not too large, a single-ended measurement will be just fine.

2.5 Noise measurements of an AD584 voltage reference

So far, we have been measuring either the shorted noise or the ground noise. We assumed the noise with shorted inputs was an intrinsic measure of the noise in the PGA amplifier and digitizer, which was consistent with what we expected.

We found a single-ended measurement had about 3x the noise as a diff measurement due to the ground bounce noise in the low-end ground lead.

It is difficult to find a voltage source with a DC voltage in the range of 1-4 V which will have less noise than the ADS1115.

One option is the AD584 band gap reference voltage. The specification for this part is that the peak to peak voltage noise in the 0.1 Hz to 10 Hz bandwidth is about 50 uV. The rms noise value at the 108 Hz sample rate might be on the order of 10 uV to 30 uV.

To measure the 2.5 V DC voltage, the most sensitive scale is with a gain of 1x. This means the LSB is 0.125 mV. If the noise is digitizing noise, we might expect about 0.125 mV/2 = 0.062 mV. Figure 2.9 shows the noise drop off with averaging of the 2.5 V reference using a differential measurement between the output and its local ground.

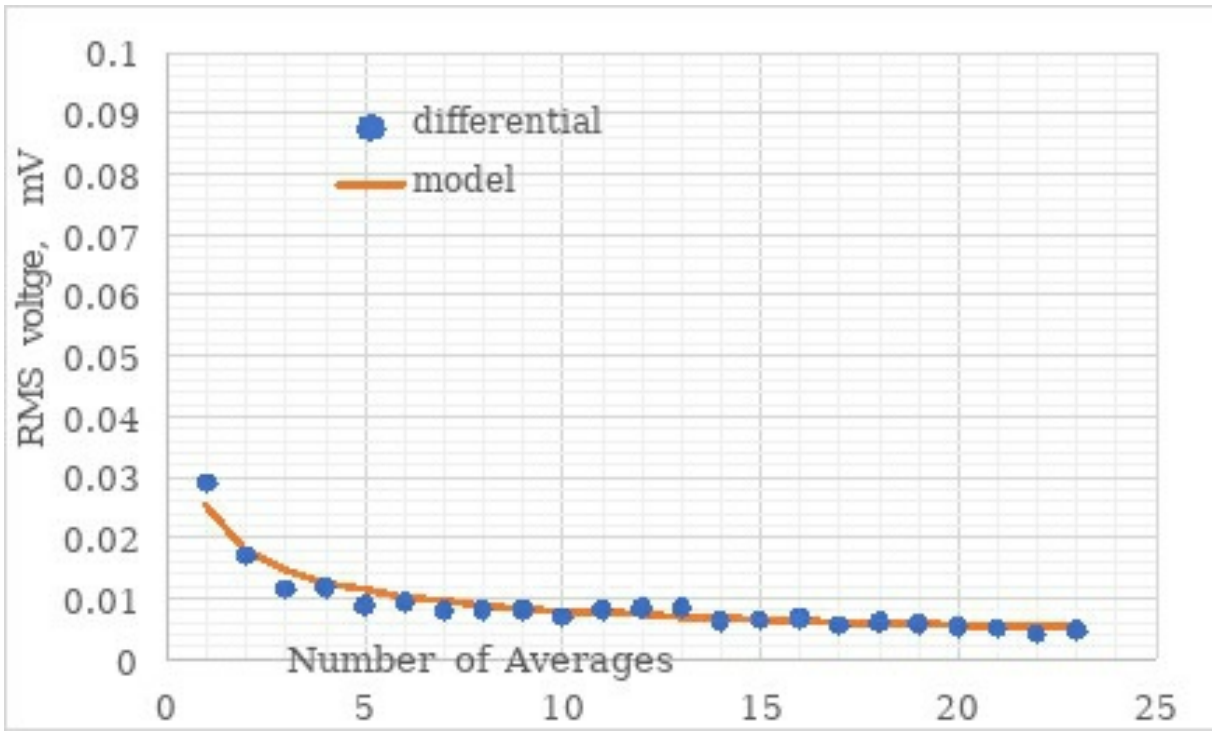


Figure 2.9 Measured noise on the AD584 reference voltage with a differential measurement of the 2.5 V output.

The model fit is with an rms voltage at 108 SPS of 0.025 mV. This is half the noise level expected based on digitizing noise. This suggests the noise we are measuring is probably digitizing noise and the intrinsic noise of the AD584 is less than this.

When we use a single-ended measurement for this voltage, we expect more noise due to the ground bounce noise on the package lead.

This is exactly what is measured. Figure 2.10 shows the rms noise on the 2.5 V reference measured as a single-ended voltage. On the same plot is the rms noise of the same source measured with a differential set up.

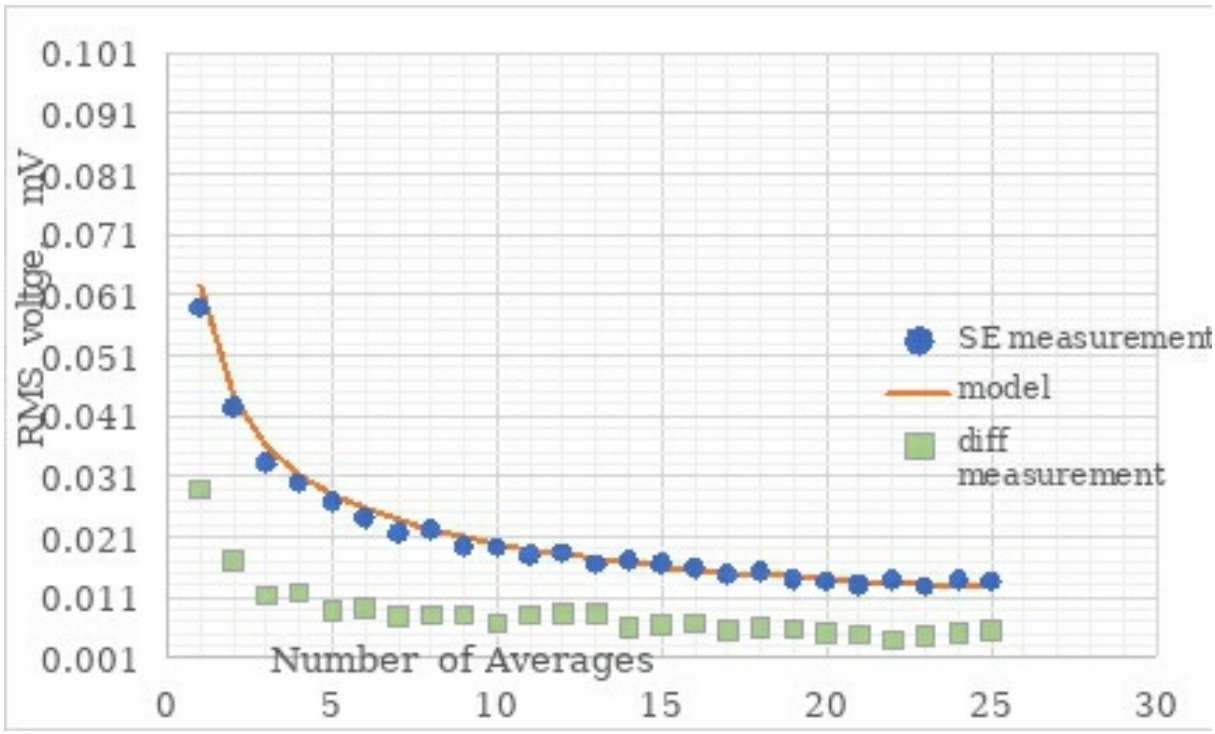


Figure 2.10 Measured noise drop off for a single-ended measurement and a differential measurement.

The model fit for the single ended measurement is an rms level of 0.062 mV. This is ½ LSB. This noise in the SE measurement is twice the noise of the diff measurement.

In addition, the DC voltage level of the 2.5 V reference signal is also different when measured with a SE or differential measurement. As we saw before, there was about 0.25 mV difference due to the DC current flow in the on-die metallization and package lead resistance. Figure 2.11 shows the difference in DC voltage measured for the SE and differential measurements. The difference is about 0.3 mV.

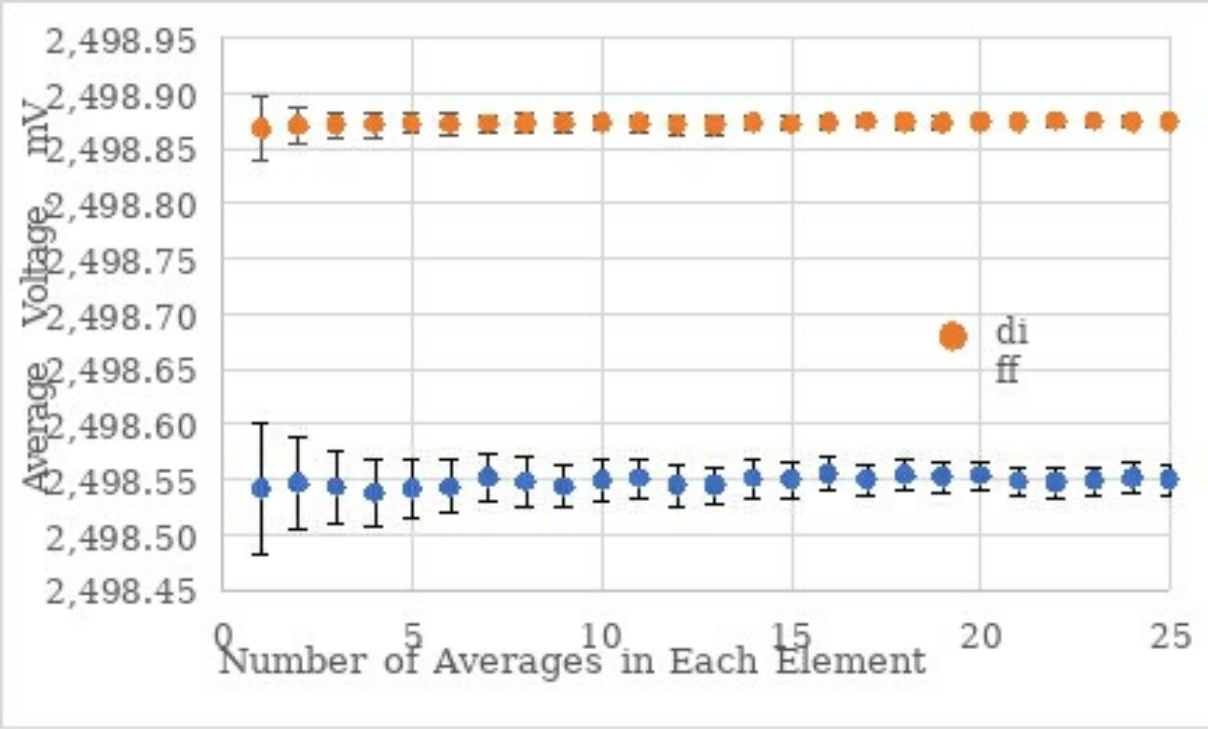


Figure 2.11 Measured DC voltage on the 2.5 V reference voltage with SE or differential measurements. The scale is 50 uV/div.

The voltage of the 2.5 V reference I measured with a Keithley 196 DMM was 2499.0 mV. The ADS1115 measured a value of 2498.9 mV. This is a difference of 0.1 mV. These were measured between the 2.5 V output and the local gnd of the AD584. The absolute accuracy is rated at only ±1 mV.

The rated accuracy of the AD584 reference voltage source is rated at ±2.5 mV. It clearly meets this specification, being within 1 mV of 2.500 V and within 0.1 mV of the Keithley 196 measurement.

It is remarkable that even with a rated absolute accuracy to within only ±1 mV, the agreement with the differential measurement of the ADS1115 and Keithley 196 is within ±0.1 mV.

2.6 Mutual-aggression noise

So far, the analysis has been on the intrinsic noise within one ADS1115 channel. This is the self-aggression noise.

Now we consider the mutual-aggression noise or the cross talk between one

channel and another channel. This is the noise picked up on one channel when there is a voltage on the second channel.

To measure this cross talk, we need to use a periodic signal in one channel and look for the corresponding pattern in another channel.

This is implemented by using a function generator to create a 2 V peak to peak voltage signal, with a 0.01 Hz square wave. This is a period of 100 sec.

I measured the voltage on each of the four channels at the rate of 108 SPS and interleaved the four channel measurements. The measurements over a 1 second period are averaged in each channel and these 1-sec averages are output into excel as a voltage vs time, V(t) value.

For the first experiment, the four individual measurement elements were:

0: measured as SE connected to gnd on A0, gain setting of 1x

1: measured as the 1 V to 3 V square wave aggressor signal on A1

2: the SE measurement of the 2.5 V AD584 reference voltage

3: the diff measurement of the 2.5 V AD584 reference voltage

A total of 1 hour of measurements, recorded every 1 sec for each of the four channels were sent to excel. The analysis was done in excel.

Here are a few observations:

The difference between the single-ended and differential measurements of the 2.5 V reference voltage is just as expected. Figure 2.12 shows this comparison. There was about a 0.3 mV difference in their DC values and the noise in the differential measurement is less than half that of the SE measurement.

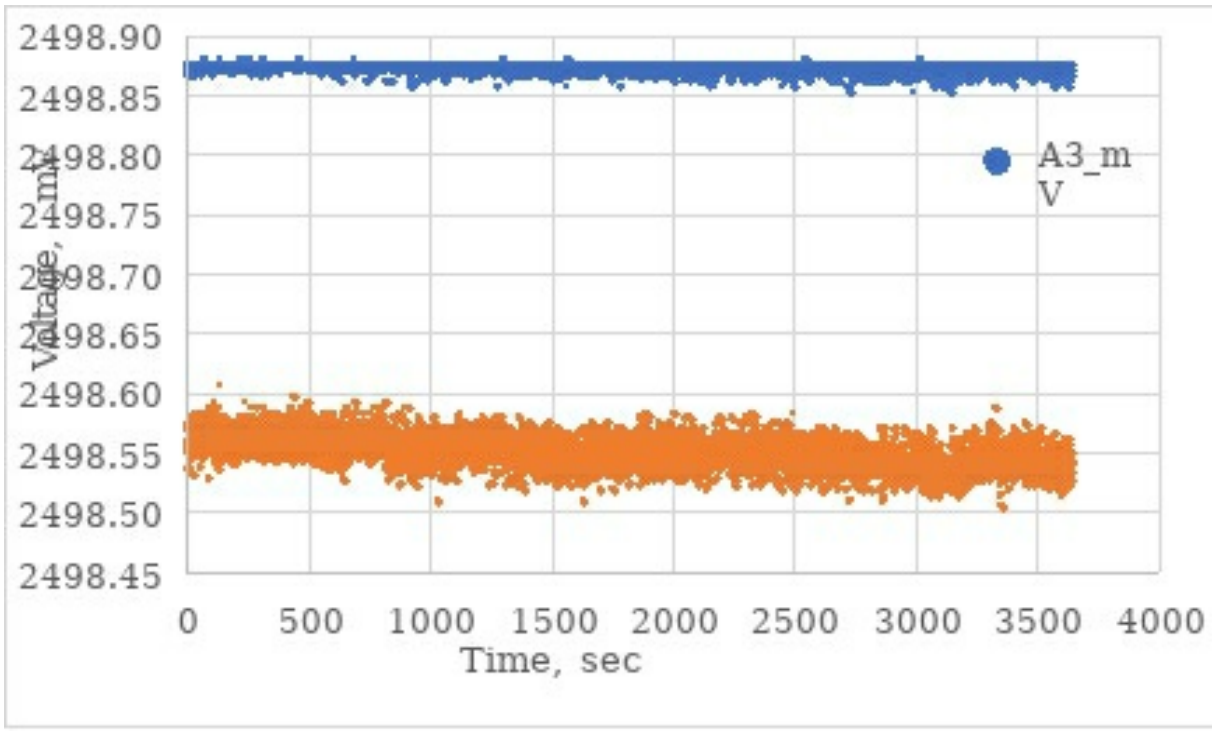


Figure 2.12 Comparison of the differential measurement and SE measurement of the 2.5 V reference voltage.

The calculated rms value for the differential measurements of the 2.5 V reference signal is 3.9 uV.

The calculated rms value for the single-ended measurements of the 2.5 V reference signal is 14 uV.

The aggressor square wave is from 1 V to 3 V with 100 second period. This signal measured as a SE signal on channel A1 is shown in Figure 2.13.

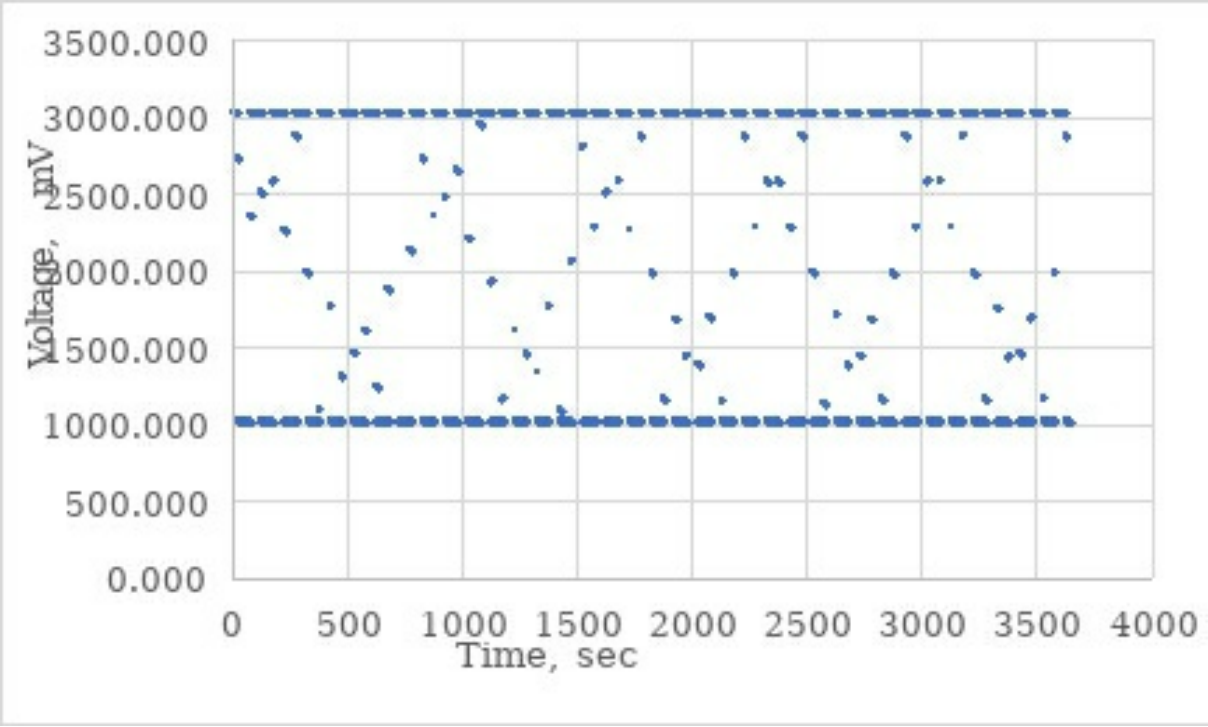


Figure 2.13 Measured voltage from the square wave generator, on channel A1.

There is no obvious 0.01 Hz voltage variation component in the diff or SE voltage measurement of more than about 20 uV.

Using this limit, the rejection of mutual aggression noise from channel to channel is about 20 uV/2 V = -100 dB.

We cannot get any better estimate by visual inspection. To improve the measurement of the residual noise at the 0.01 Hz amplitude of the aggressor, we have to use a numerical calculation.

2.7 Calculating the first harmonic component

We can calculate the amplitude of the frequency component at the 0.01 Hz frequency in the aggressor signal and in the victim channel. This will be a better metric of the isolation and mutual-aggression cross talk.

This is done by calculating the amplitude of the harmonics for a fixed period, T0:

$$A_n = \frac{2}{T_{total}} \int_0^{T_{total}} V(t) \sin\left(n \frac{2\pi}{T_0} t\right) dt$$

and

$$B_n = \frac{2}{T_{total}} \int_0^{T_{total}} V(t) \cos\left(n \frac{2\pi}{T_0} t\right) dt$$

This is the discrete Fourier Series; we will only have to calculate one harmonic for n = 1.

The assumption in the discrete Fourier Series is that we integrate over an integer number of cycles. However, when we start the measurement, the phase of the square wave is arbitrary. Likewise, while the period is 100 seconds, we are not taking measurements with an interval between points that is exactly 1 second. The time interval between consecutive measurements is actually 1.011 seconds per point. The total time to take the 3600 points is not exactly 3600 seconds.

This means we do not end after an integral number of cycles. To correct for this, I deleted the first set of data until the measured square wave just turned off from 3 V to 1 V and I deleted the last set of data so it ended on the square wave just turning off from 3 V to 1 V. This guaranteed we have an integral number of cycles in the dataset of 3558 final samples, and the integral is over an integral number of cycles.

As a check on this, we calculated the 1st harmonic amplitude of the square wave aggressor signal.

In a square wave, the amplitude of the first harmonic is ½ x peak-to-peak x 4/pi. In the case of the aggressor, the peak to peak value was measured as 2014 mV. This means the amplitude of the first harmonic at a period of 100 seconds, is

$$A = \frac{1}{2} V_{pp} \frac{4}{\pi} = 2014 \frac{2}{\pi} mV = 1282 mV$$

Applying this integral to the reduced dataset of 3558 measurement points, which is an exact integral multiple number of cycles, I calculated a 1st

harmonic of the measured aggressor signal of 1281.3 mV. This is within 0.7 mV or 0.05% of what we expected to see.

I implemented one additional step to reduce the numerical noise due to the multiplications and integrals. We are looking for a very small change out of the roughly 2500 mV DC signal level.

In a sine or cosine wave, the average value of an integral number of cycles is 0. In principle, the average of an ideal sine wave with some large amplitude, should still be zero. But, with a large DC value, we are adding together many large plus and minus numbers and looking for a very small residual number. Dealing with the large values can introduce some numerical noise.

To reduce this problem, I calculated the average of each set of 3558 data points and then subtracted this average from each voltage that was multiplied in the integral. This way, the actual voltage value multiplied by the sine or cosine term in the integral was a small value. This dramatically reduced the numerical noise. The integrals become:

$$A_n = \frac{2}{N}\int_0^{T_{total}} \left(V(t) - \langle V(t)\rangle\right)\sin\left(n\frac{2\pi}{T_0}t\right)$$

and

$$B_n = \frac{2}{N}\int_0^{T_{total}} \left(V(t) - \langle V(t)\rangle\right)\cos\left(n\frac{2\pi}{T_0}t\right)$$

I also substituted the total number of data points measured, N = 3558.

This was the integral performed on the square wave data to reach the 0.05% agreement with the expected first harmonic value of the idea square wave.

Using this analysis, the first harmonic of the SE measurement of the 2.5 V reference voltage, at the 0.01 Hz frequency of the aggressor was extracted as 0.7 uV amplitude. The first harmonic of the differential measurement was extracted as 0.2 uV.

Using the 0.7 uV amplitude, this is a channel to channel DC cross talk at 0.01 Hz of 7×10^{-8} V/1.2 V = 6×10^{-8} or -144 dB of channel to channel isolation.

This is a remarkable performance for such a simple and low-cost system.

2.8 Summary of the noise specs of the ADS1115

The intrinsic rms noise from the ADS1115 is about ½ LSB at the highest sample rate of 108 SPS. This is about 3.9 uV rms on the gain 16x scale.

This rms value will decrease with the square root of the number of averages.

A single-ended measurement will more than double this noise level due to the ground bounce in the ground path from the external board to the internal pad on the IC connected to the PGA low input.

The absolute voltage accuracy of the ADS1115 using a differential measurement is better than the rated accuracy of most high-end system voltmeters, better than ±1 mV absolute accuracy.

The channel to channel isolation is better than -140 dB.

If noise and accuracy is important, a differential measurement should be used.

In a single-ended measurement, there may be a voltage offset of as much as 0.3 mV.

Given these levels of noise, we can evaluate how the voltage of various voltage source vary over time.

Chapter 3. Voltage Stability of Voltage Sources

When we measure the ADS1115 over a 1 second interval, there will be 108 measurements. If these are split over 4 channels, there will be 27 measurements per channel.

If the voltage scale is ±4.096 V, the rms noise in the 108 SPS measurements will be about 0.125 mV/2 = 60 uV. With 27 measurements in each channel, for each 1 second interval, the noise will be about 60 uV/sqrt(27) = 11 uV.

We measured 14 uV in a single-ended measurement and 3.9 uV in a differential measurement looking at the AD584 reference.

We can look at the DC voltage variation of any source with a 1 sec averaging time and over a long period of time as either a SE or differential measurement.

The sketch I used takes 4 consecutive channel measurements. Each channel can be configured with any scale and any combination of single ended or differential. These measurements are measured interleaved over a fixed time interval, typically 60 power line cycles or 1 second, and their average recorded and printed into an excel spreadsheet.

With this tool we can measure the important figures of merit for a variety of different sources:

- ✓ *the absolute voltage levels*
- ✓ *how much they drift over 1 hour*
- ✓ *their rms noise*

.

3.1 The AD584 voltage reference

The Analog Devices AD584 is a precision voltage refence IC with three different output voltages: 10 V, 5 V and 2.5 V, accurate to better than 0.1% absolute accuracy. It can be purchased for about $1 each from here.

The datasheet can be downloaded from here.

Here are some important features from the datasheet:

1. *If powered by more than 12 V, there are three output voltages available: 10 V, 5 V and 2.5 V.*
2. *If powered by 5 V, the 2.5 V is still available. By connecting the 2.5 V output pin to the 10 V output pin, the 2.5 V is a buffered output.*
3. *It can source 10 mA of current and sink 5 mA of current.*
4. *On the 2.5 V range, the output voltage is accurate to ±3.5 mV. This is about 0.1%.*
5. *The peak to peak noise in the range 0.1 Hz to 10 Hz is 50 uV.*
6. *The long-term stability over many hours is 25 ppm. For the 2.5 V rail, this is 62 uV.*
7. *The temperature impact is 20 ppm/degC. This is about 50 uV per deg C change.*

When all we require is a 2.5 V reference, we can power the AD584 from 5 V and connect pin 3 to pin 1 which creates a buffered output of 2.5 V. This is shown in Figure 3.1.

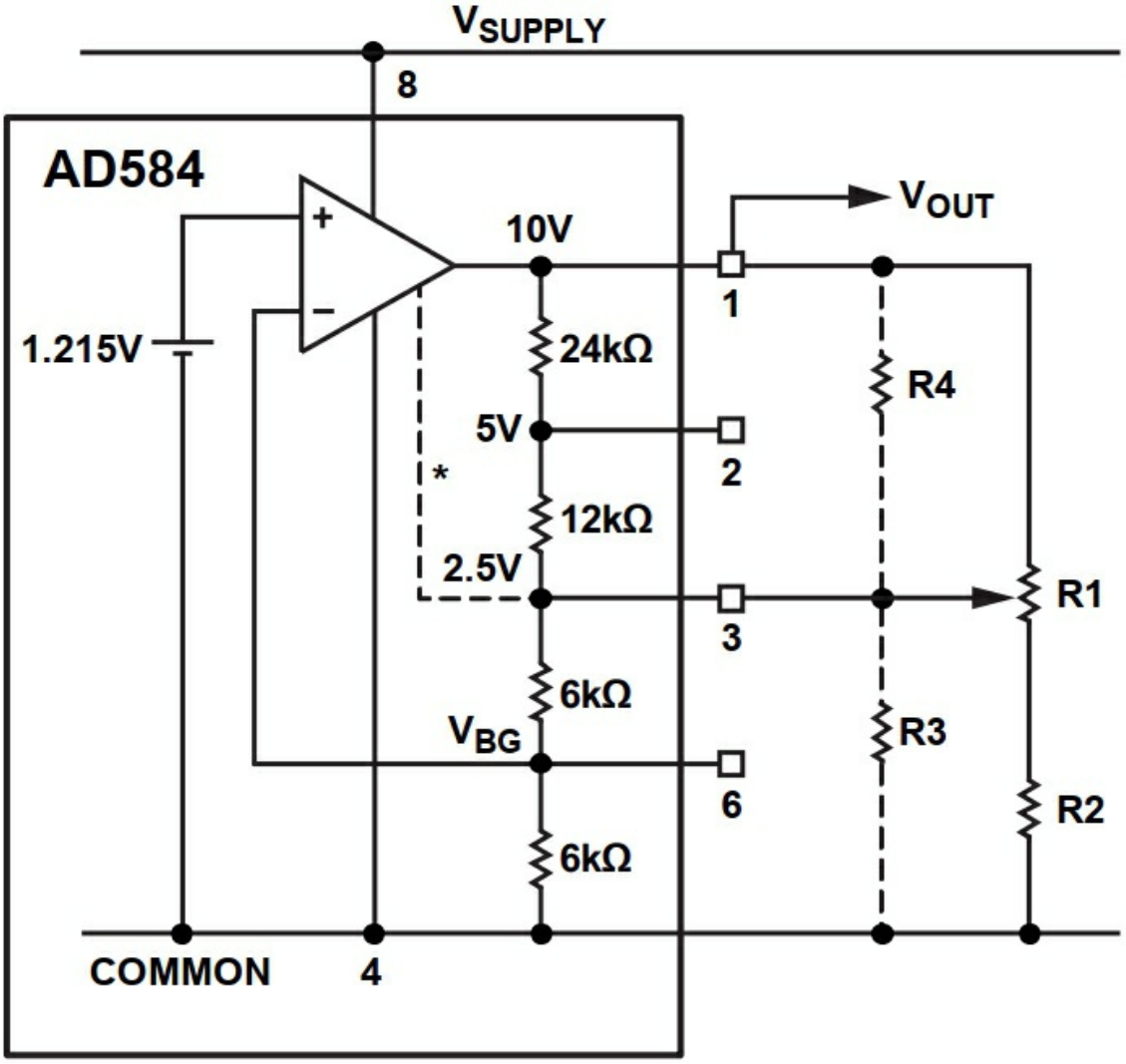


Figure 3.1 Circuit diagram of the AD584 from the datasheet. When pin 3 is connected to pin 1, the output is the buffered 2.5 V reference, used in all these measurements.

When the AD584 output was measured over an hour period of time, there is no discernable drift and the rms noise level is below the expected rms noise level.

As an example, Figure 3.2 shows the measured single-ended and differential measurement of the 2.5 V reference, with 1 second averages, on the gain setting of 1x and 27 measurements per point. The expected rms voltage is based on ½ LSB with 27 averages over the 1 second interval or about 0.5 x 125 uV/sqrt(27) = 12 uV rms.

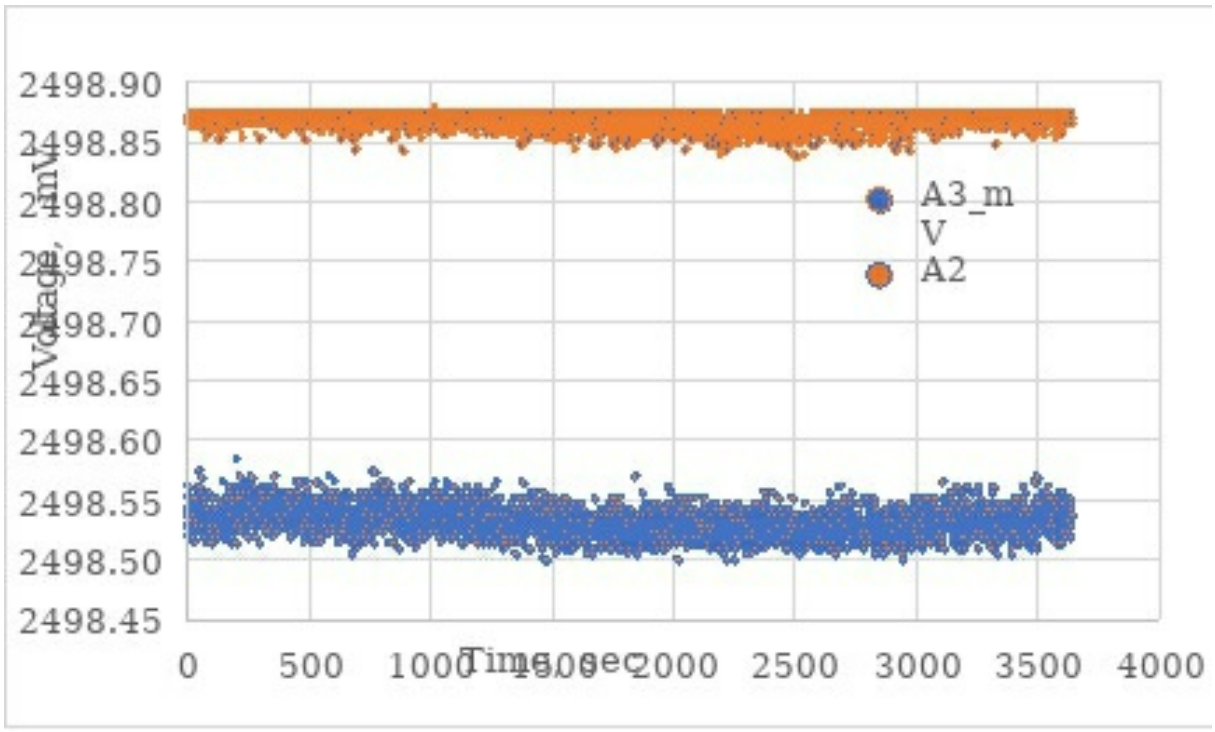


Figure 3.2 Measured output of the AD584 over 1 hour. Top is differential and bottom is single-ended measurement. The vertical scale is 50 uV/div.

The calculated rms value for the SE measurement is 12 uV and for the differential measurement, 7 uV. These are both in line with what is expected.

This confirms that the AD584 is a very stable source and that the ADS1115 is a very stable measurement system.

Further, the voltage of the AD584 was measured as 2.499 mV with a Keithley 196 DMM while it was also being measured by the ADS1115. The agreement between these voltages is within 0.15 mV.

This says the absolute accuracy of the ADS1115 is accurate to better than ±1 mV and the absolute accuracy of the 2.5 V reference voltage of the AD584 is better than the specified value of ±2.5 mV.

These measurements suggest that the performance of the ADS1115 and AD584 are very consistent. There are both very stable to within 20 uV over a period of an hour.

The drift in the AD584 is less than 20 uV over an hour.

The rms voltage noise of the AD584 is 11 uV when measured SE and 7 uV

when measured as a differential signal with external ground. Both of these measurements could be within the noise floor of the ADS1115.

3.2 Experiment: Two independent AD584 references

Now that the performance of the ADS1115 has been established and the performance of the AD584 voltage reference has been established, we can do a number of experiments to explore the voltage stable and noise of a variety of voltage source.

As a first experiment, we can compare the voltage difference between two independent AD584 voltage reference chips.

To do this requires a little more room than can fit on a simple mini instant-shield. I rebuilt a different ADS1115 on a solderless breadboard, adjacent to a Sparkfun Redboard. Included on this solderless breadboard are the following components:

- *An ADS1115 module*
- *Two independent AD584 2.5 V reference chips*
- *A TMP36 temperature sensor*
- *A 1k resistor pot connected to the 2.5 V output of the AD584*
- *Two 1k resistors forming a voltage divider with the 5 V rail from the Arduino*
- *A simple 3-digit DMM to read the voltage on the 5 V rail*
- *The connections from the Arduino to the solderless breadboard:*
    - *5 V*
    - *Gnd*
    - *SCL*
    - *SDA*

This new module is shown in Figure 3.3. The performance of this module, with completely different parts, but running the same sketch is identical to the performance of the other ADS1115 module.

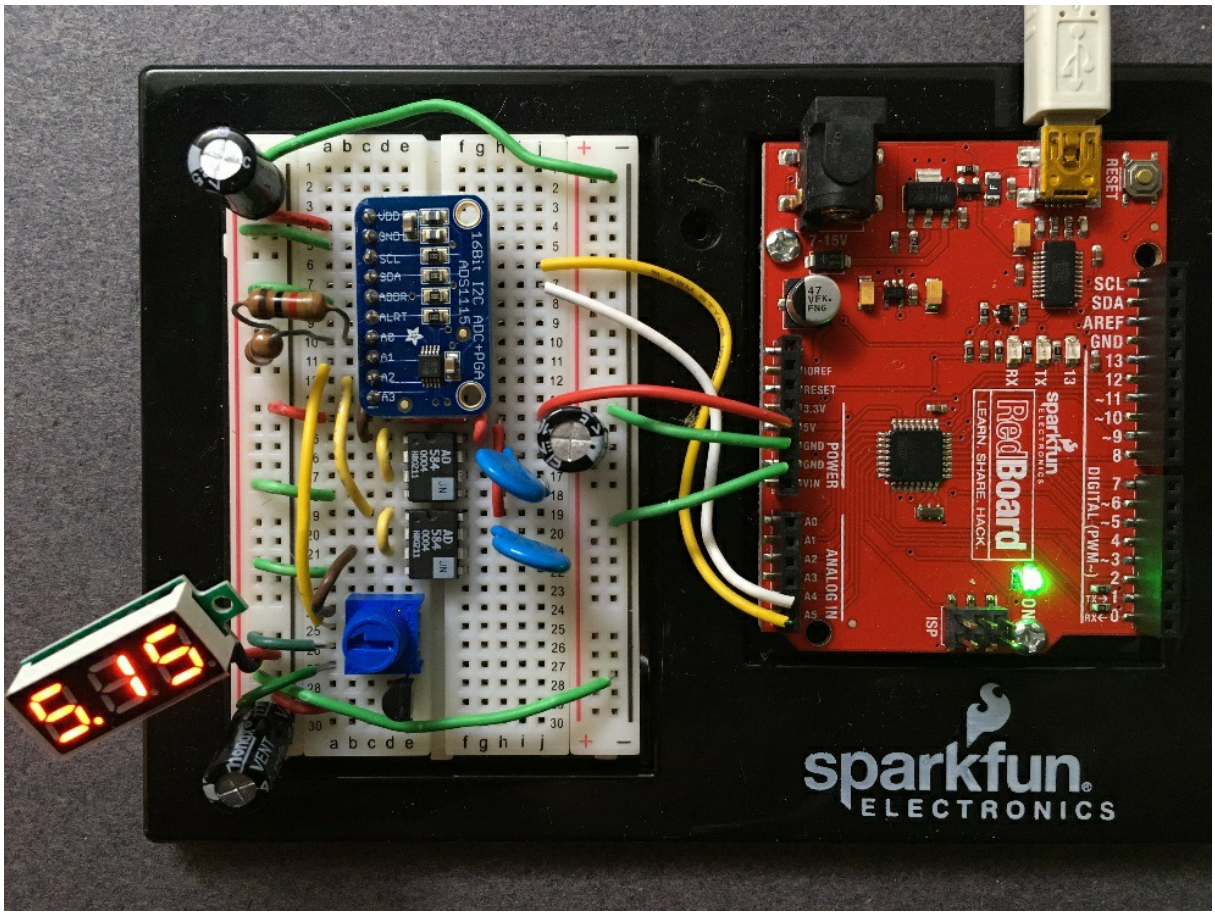


Figure 3.3 A second module built in a solderless breadboard for more experiments.

In the first experiment, the two different AD584 are used as inputs to A2 and A3 and the differential voltage is measured. Since this difference should be on the order of 2 mV, the highest sensitivity scale is used with a gain of 16x.

Figure 3.4 shows the measured difference voltage for two independent AD584 sources, with a gain of 16x. The intrinsic digitizing rms noise expected is about ½ x 7.8 uV/sqrt(27) = 0.8 uV.

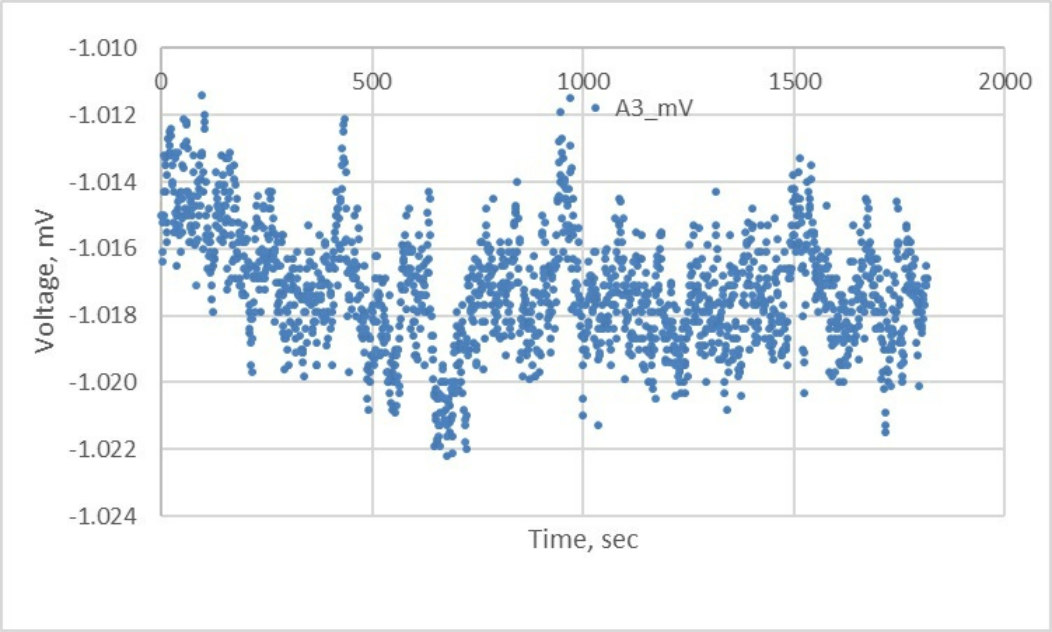


Figure 3.4 Measured voltage between two independent AD584 voltage references as a differential measurement, over a period of 30 minutes. The scale is 2 uV/div.

The absolute voltage difference between the output voltage of these two ADS584 reference sources is about 1 mV. This is well within the rated absolute accuracy of ±3.5 mV.

The vertical scale is 2 uV/div. On this scale, there is some slow variation with an overall peak to peak value less than 10 uV. This is the voltage noise between two independent AD584 sources. It is above the noise floor of the ADS1115 and represents the true voltage variation of the AD584.

If their noise adds with the square root of the sum of the squares, as uncorrelated random noise sources would, we would expect the peak to peak variation in either one to be about 10 uV/ sqrt(2) = 7.1 uV of peak to peak noise. This is still way below the 50 uV specification for the AD584 in the 0.1 Hz range.

Realistically, as a rough rule of thumb, once the AD584 has stabilized after a few minutes of being on, its output voltage can be expected to be stable to within 10 uV. This includes the 2 uV rms noise, the 7 uV peak to peak noise and any drift noise.

For $1, this is a remarkably stable source.

It is also remarkable that this level of 10 uV of noise is achievable in a

solderless breadboard with wires flopping in large loops. There are two reasons these measurements are not sensitive to 60 Hz ac pick up. First is the low impedance of the sources. This means the interconnects are sensitive to magnetic field pick up, not electric field pickup.

Second, we are averaging each measurement point over 60 power line cycles. This will digitally filter any specific 60 Hz noise. This principle is reviewed in HackingPhysics Journal, vol 1 no. 1 published in Jan 2020.

As a consistency test, another pair of AD584 references were measured in the same way, over a period of 1 hour. These measurements are shown in Figure 3.5.

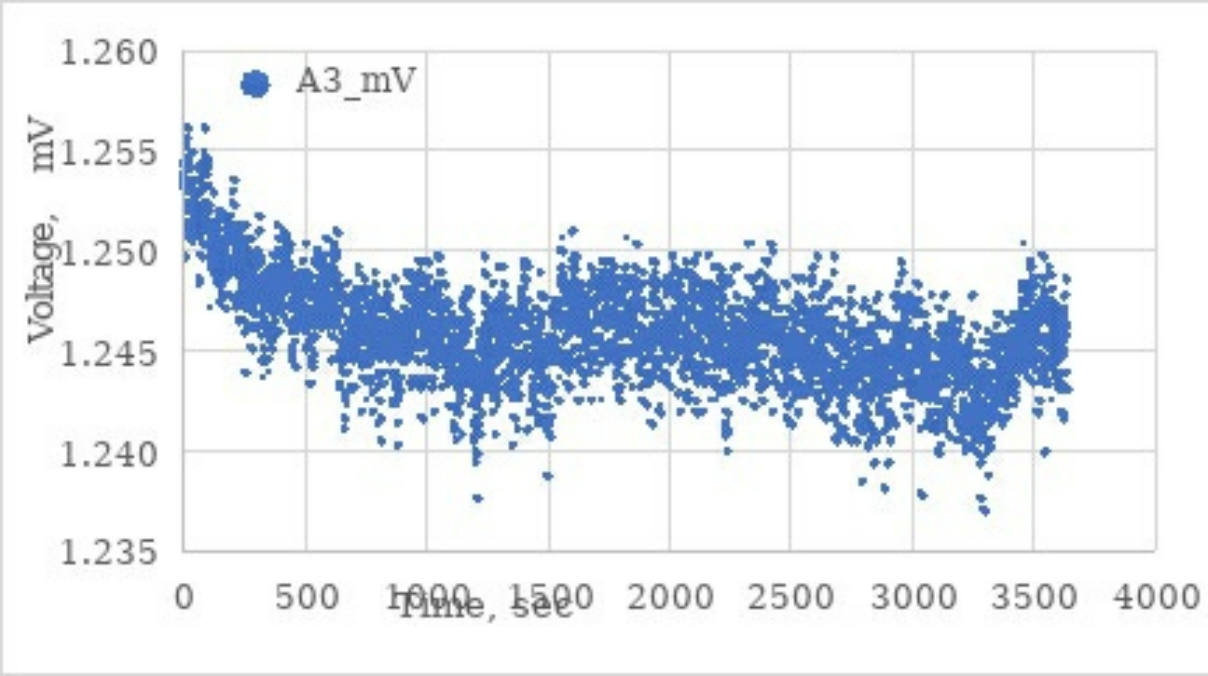


Figure 3.5 Measured voltage difference between two different, independent AD584 2.5 V reference chips. The vertical scale is 5 uV, out of a DC value of 2.5 V. This is 2 ppm/div.

The voltage variation shows a small 10 uV drift for the first 10 minutes and a stable reading with a peak to peak variation of about 10 uV. This behavior is very consistent with the previous measurements.

We can expect a voltage stability roughly within a 20 uV peak to peak range. It will be within a 10 uV range after a few minutes of settling time.

3.3 Experiment: An external voltage source from a function generator

In this example, the Stanford Research Systems DS335 function generator was used as a voltage reference. The amplitude was set as 0 V and the offset was set as 1 V peak to peak. Into a high impedance, this is a 2 V peak to peak voltage. This should be a constant voltage reference.

This voltage was measured both as a single ended voltage and a differential. The measured voltages over a 1-hour period are shown in Figure 3.6.

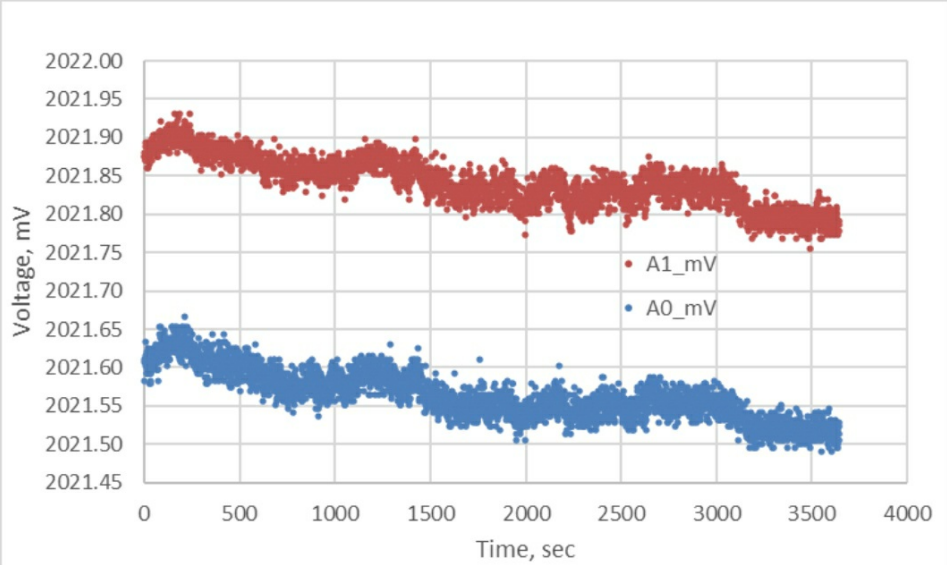


Figure 3.6 Measured output voltage for an external 2 V source. The (top) red measurement is differential, and the (bottom) blue is SE. The scale is 50 uV/div in each plot.

The absolute voltage accuracy is off by about 20 mV out of 2 V or 1%. This is as good as can be expected.

In addition, there is a slow drift of about 100 uV over the 1-hour period. Overall, for a source not meant to be accurate, the voltage is stable to within 100 uV over the 1-hour period.

3.4 Experiment: An AA battery

An alternative stable voltage source is a battery. With a voltage of 1.6 V, we can use the ±2.048 V range which is a gain of 2x in the PGA. This will

reduce the noise floor slightly compared with the gain of 1x scale.

Two different AA batteries were measured, each as the differential voltage between its positive and negative terminals. This gets around the ground bounce noise in the lead of the ADS1115. This measurement system is shown in Figure 3.7.

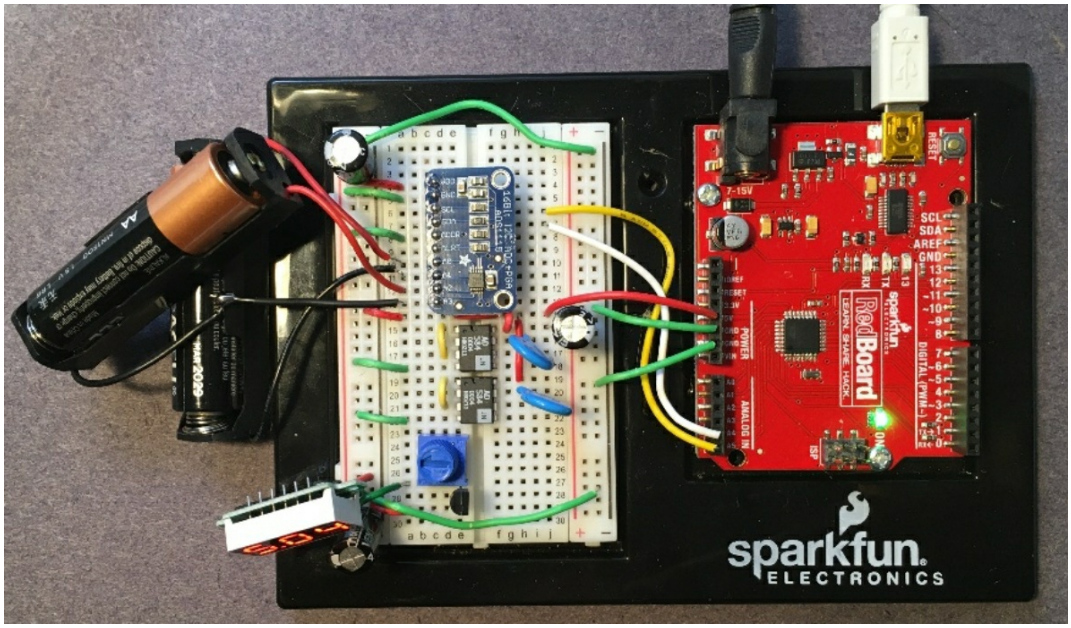


Figure 3.7 The measurement configuration for two AA batteries as differential measurements between their positive and negative terminals.

One of the batteries had been used with a current draw of about 1 mA for a few minutes, and then measured. The other one had been unconnected to any load. Figure 3.8 shows the different behavior of these two batteries.

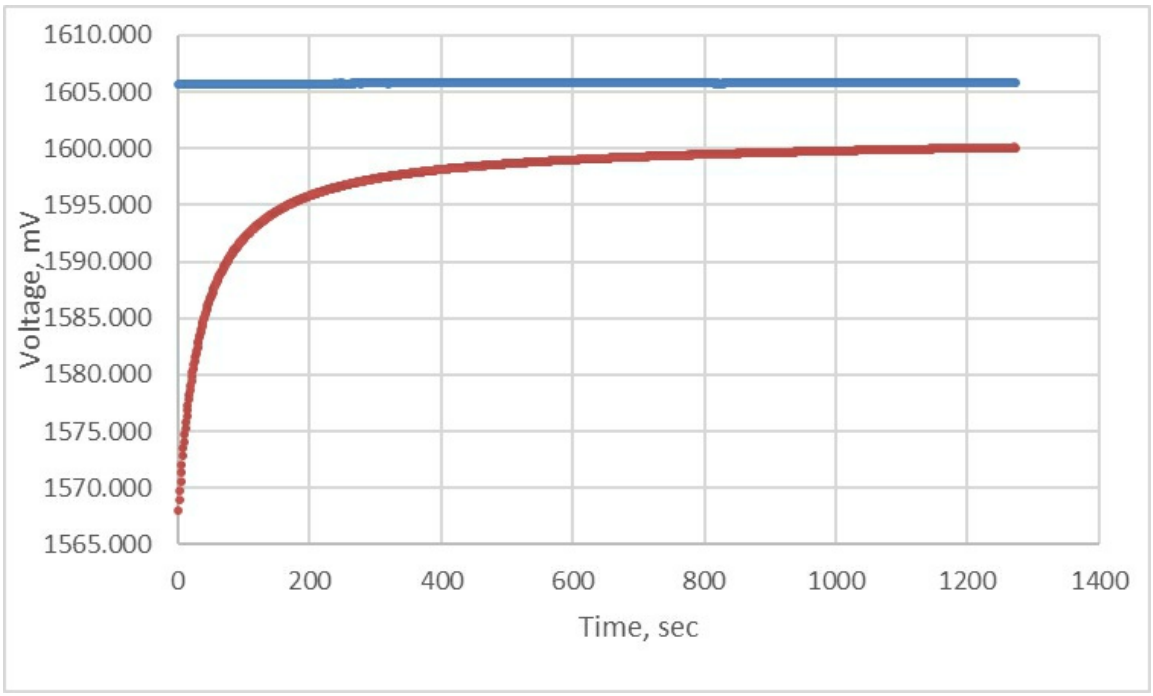


Figure 3.8 The voltage behavior of a fresh, unused AA battery (top, blue) and a battery that had recently been used for a few minutes at 10 mA discharge current (bottom, brown). The scale is 5 mV/div.

The battery that had been previously used shows a long settling time for its voltage to stabilize. During the period of observation, it changed about 30 mV. This suggests a battery is not a good stable reference voltage unless it is fresh and has not been used for at least a day.

If it has supplied any current, its output voltage will slowly creep back up in value and this equilibration time may be many hours.

A fresh, unused battery has a short-time noise floor close to the ADS1115 limit. However, it is also subject to some slow drift. Figure 3.9 shows the measured voltage on the good battery that had not been used.

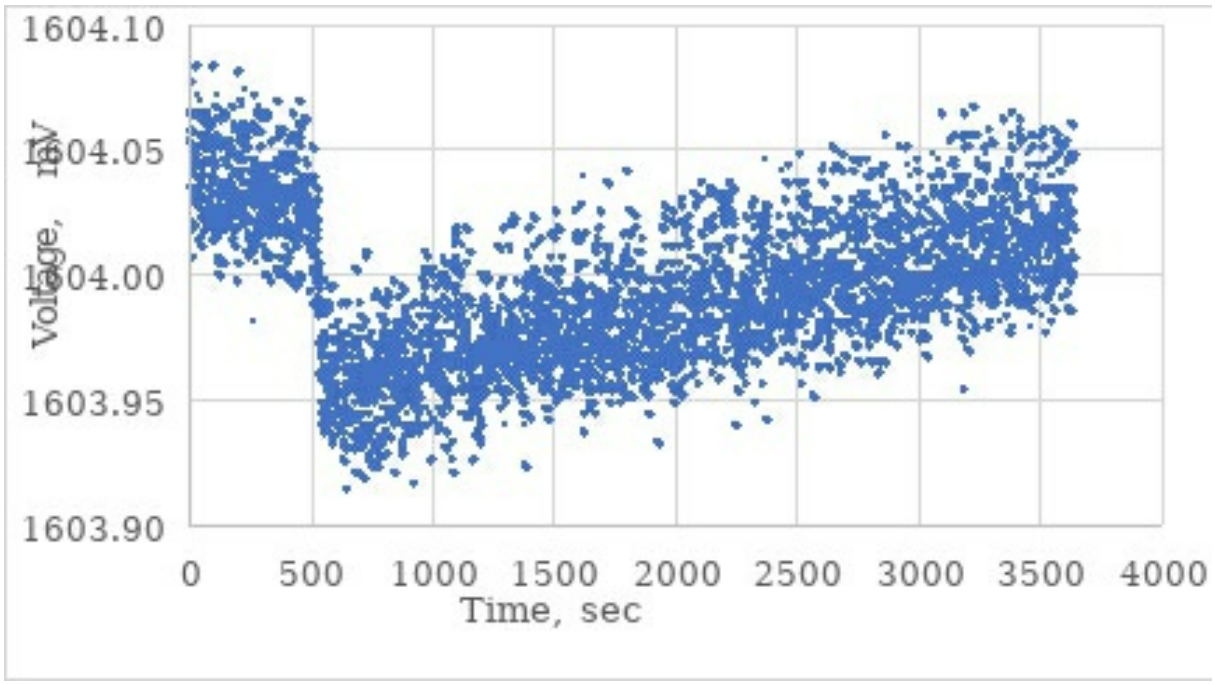


Figure 3.9 The voltage on an unused battery over a 1-hour period. The scale is 20 uV/div. At the 500 second mark, I touched the battery very lightly.

The short time noise is on the order of 80 uV peak to peak. The DC absolute battery voltage is also very sensitive to temperature. To illustrate this, at the 500 second mark, I touched the battery very lightly for 5 seconds.

This was enough to heat the battery very slightly and shift the cell voltage. The electrochemical cell voltage of this alkaline AA battery decreases with an increase in temperature. The voltage change from the slight temperature rise was only about 50 uV, but easily measured with this instrument.

Note the rapid time for the temperature to decrease the cell voltage due to the forcing function of my warm finger and the very slow time for the voltage to increase due to the cooling of the battery.

This is why a battery should never be used as a voltage reference. In addition to its voltage very dependent on its previous usage, it is also very sensitive to the ambient temperature and when it was last handled.

3.5 Experiment: 5 V USB hub

An important feature of the PGA on the front end of the ADS1115 is that the

gain can be set for TWO-THIRDS. This means a voltage as high as 6.144 V can be measured, even though the Vdd level is only 5 V. This is perfect for measuring a 5 V rail supplying power to the ADS1115.

The 5 V rail supplied by a USB hub, was measured directly with one channel of the ADS1115 as a single-ended measurement on the scale of TWOTHIRDS. Figure 3.10 is the measurement over a 1-hour period.

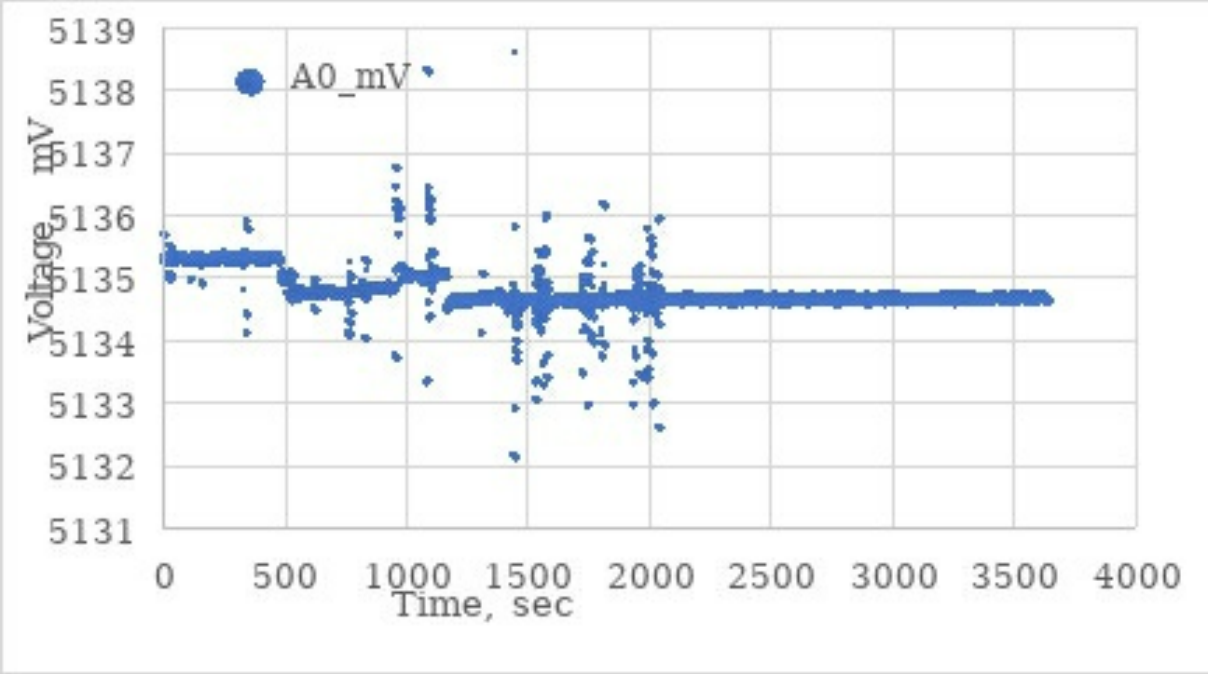


Figure 3.10 Measured voltage on the 5 V USB hub. The gain of the ADS1115 is 2/3 x. The scale is 1 mV/div.

During the measurement, I touched the Atmega 328 uC on the Arduino board. This changed its temperature which changed the current draw of the uC, which changed the DC voltage on the 5 V rail on the board. At the level of 1 mV of voltage stability, everything affects the voltage on the Arduino board from the USB hub.

The typical value is about 5.135 V. But there are fluctuations on the order of 5 mV peak-to-peak variation. In extreme cases, there are pulses of voltage noise with 5 mV excursions. Each measurement in this example is an average over 1 sec intervals.

When voltage stability is important, don’t use the 5 V from the USB hub.

3.6 Experiment: 5 V rail from the LDO

An alternative source of 5 V rail on the Arduino board is the on-board LDO. This powers the 5 V rail only when an external power source of 7-12 V is plugged into the power jack.

The on-board LDO is activated when a high enough voltage is plugged into the power jack. Typically, an LDO provides a very stable voltage, accurate to within 1% and stable.

In this experiment, an external 12 V AC to DC converter was plugged into the Arduino board. The same measurement with the ADS1115 with a gain of 2/3 was used. Figure 3.11 is the measured voltage on the 5 V rail over a 1-hour period.

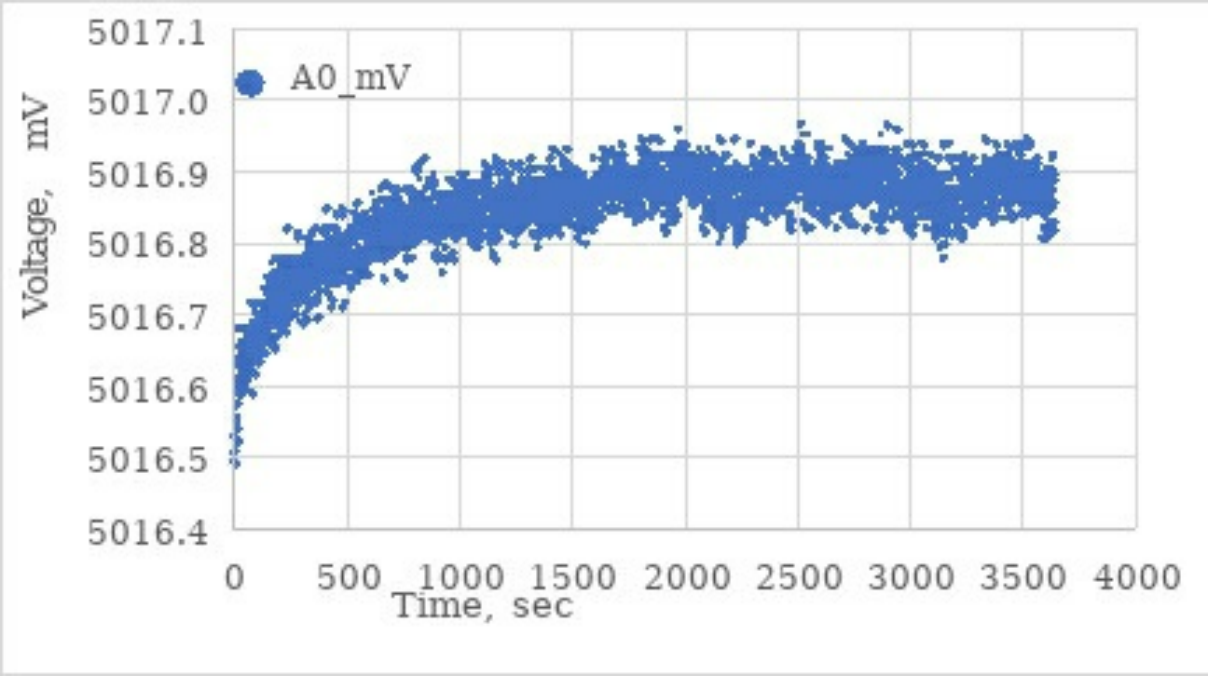


Figure 3.11 Measured voltage noise on the LDO supplying 5 V to the Arduino board. Scale is 0.1 mV/div.

The variation in this LDO is much less than for the USB hub 5 V rail. There is an initial 0.5 mV of slow transient voltage adjustment with a time constant of about 10 minutes.

After this initial period, the voltage is stable to about 100 uV peak to peak.

The absolute voltage of the LDO is 5.013 V. This is 5 V to within about

0.2%, though the rated accuracy is only 1%.

If stability of 1 mV is acceptable, the 5 V rail supplied by the LDO when powered with an external power supply is a perfectly acceptable voltage source.

3.7 Experiment: A resistor pot and the 5 V LDO as a reference offset voltage

The 5 V from the USB hub that powers the Arduino is not always a good voltage source, depending on the USB hub or computer. A more stable option is to use an external 12 V power supply to power the Arduino board and take advantage of the 5 V Low Voltage Dropout (LDO) voltage regulator on the Arduino board.

We combine a small 1k resistor pot using the 5 V rail from the LDO as the reference to generate a variable voltage from 0 to 5 V. We can evaluate how stable this adjustable voltage from the pot is by measuring it directly as a single ended voltage, and the difference between this variable voltage and the 2.5 V output from an AD584 in a differential measurement.

The configuration for this experiment is shown in Figure 3.12.

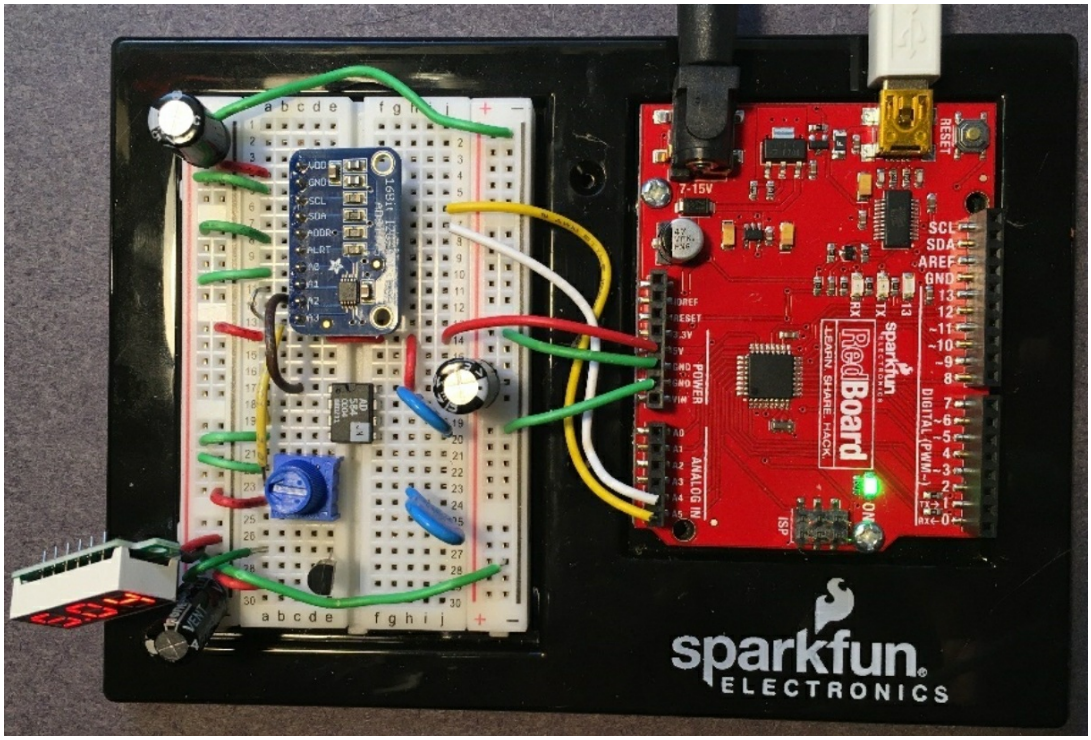
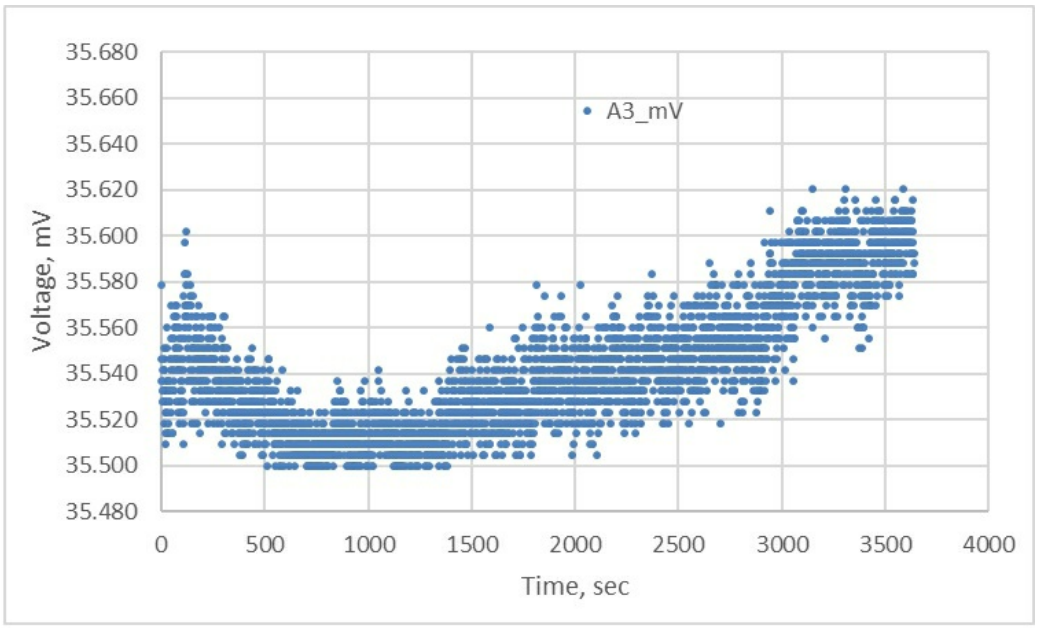


Figure 3.12 Measurement set up to use the 5 V LDO as the reference on the blue 1k resistor pot offsetting the AD584 reference voltage.

I adjusted the pot value to give a nearly zero output on the differential signal. This nulls out the 2.5 V of the AD584 reference source with the adjustable voltage using the 5 V LDO adjustable voltage.

After waiting a few seconds for any mechanical transients to die down in the resistor pot, I recorded the absolute voltage on the pot, which is related to the 5 V LDO voltage and the voltage difference of the pot value and the AD584 voltage. This long-term plot is shown in Figure 3.13.



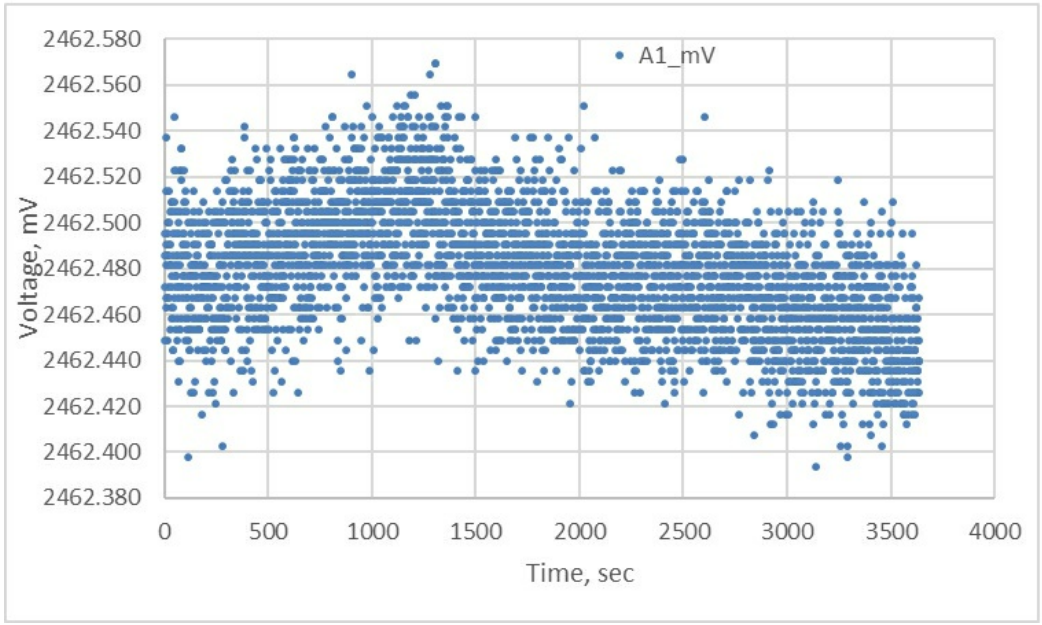


Figure 3.13 Measured voltage drift in the differential signal between the AD584 and adjustable pot (top) and in just the adjustable pot voltage (bottom). Note the scale is 20 uV/div for both plots.

This shows that for a reference voltage within a 200 uV range over an hour, using the 5 V LDO supplied power rail and a resistor pot is a viable approach to produce a stable voltage that can be used to offset another voltage using a differential input.

Note, this shows that when doing a differential measurement, as in the top

figure, the noise level is about half that of a single ended measurement. This is because of the additional noise on the internal ground reference point inside the ADS1115 in a single-ended measurement.

For voltage references with stability on the order of 200 uV, this is a perfectly fine way of providing an offset voltage. Once set, it is stable over a long period of time.

Initially, when I began measurements, I noticed there was sometimes a slow drift of about 1 mV in the output of the resistor pot. This is only about 0.04% variation, still remarkably stable. An example of this slow drift in the output voltage from the resistor pot is shown in Figure 3.14.

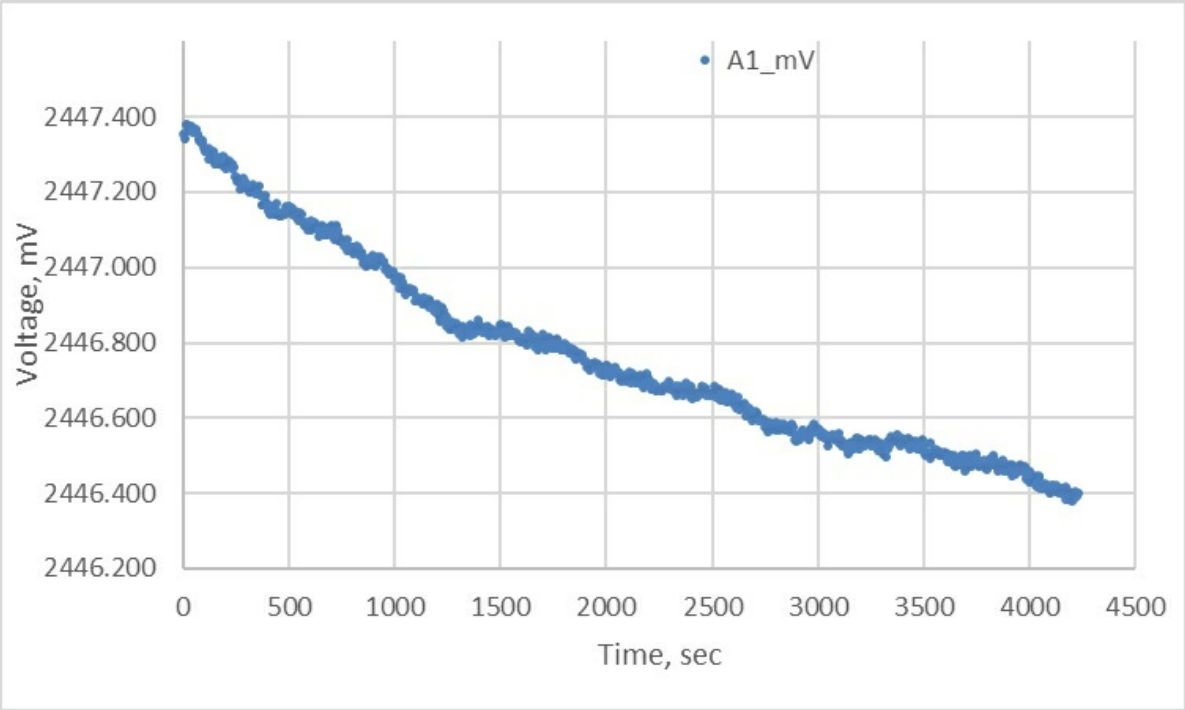


Figure 3.14 An example of the measured voltage across the resistor pot connected to the 5 V supplied by the LDO. Note the scale is 200 uV/div.

As a comparison, Figure 3.15 shows the voltage measured for the 2.5 V AD584 output voltage and the 5 V LDO adjusted with a pot to be close to 2.5 V.

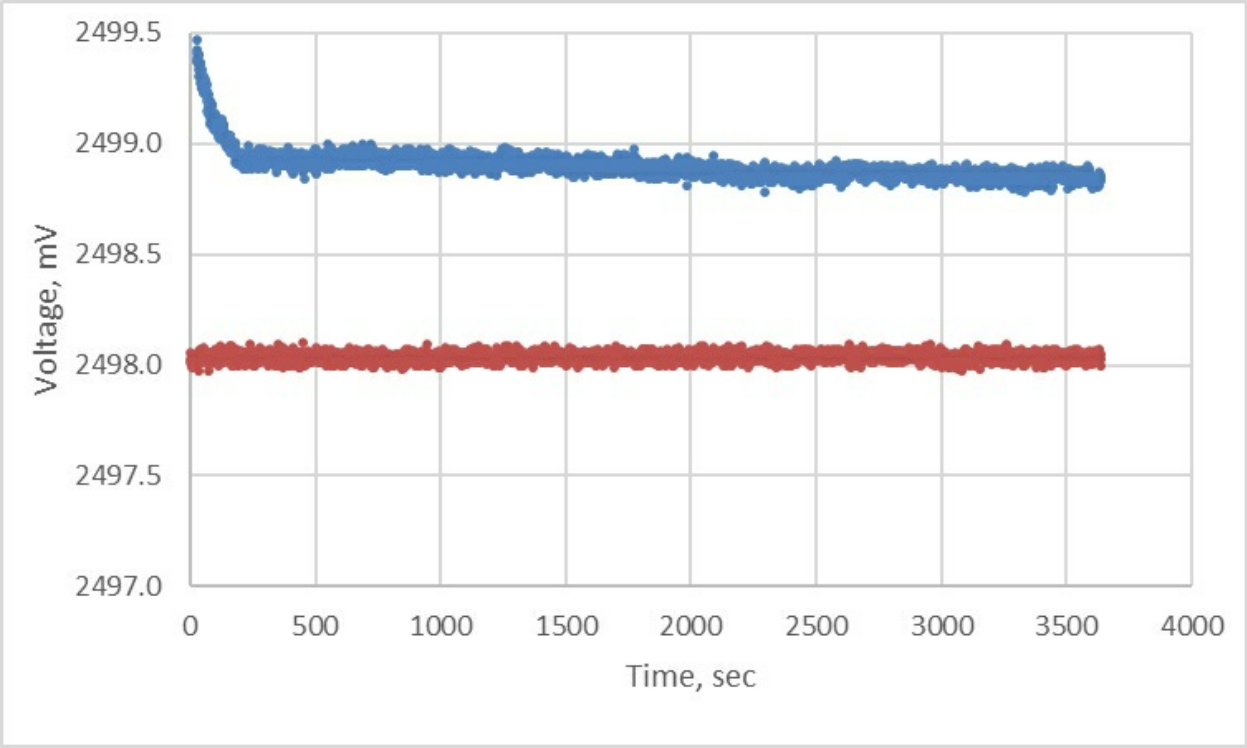


Figure 3.15 The measured voltage from the AD584 reference source measured directly (bottom in brown) and the pot from the 5 V LDO reference (top in blue).

This experiment suggests that for the most stable reference voltage the AD584 voltage reference should be used. The second-best option is the 5 V LDO on the Arduino board.

An adjustable voltage reference can be created using these voltage sources and a resistor pot. It's just important to wait for the transient mechanical relaxation of the resistor pot to settle down, and for stability better than 1 mV, don't tap the pot.

I noticed that after making a small change to the pot position, there was an initial transient in the measured voltage. This is shown in Figure 3.16.

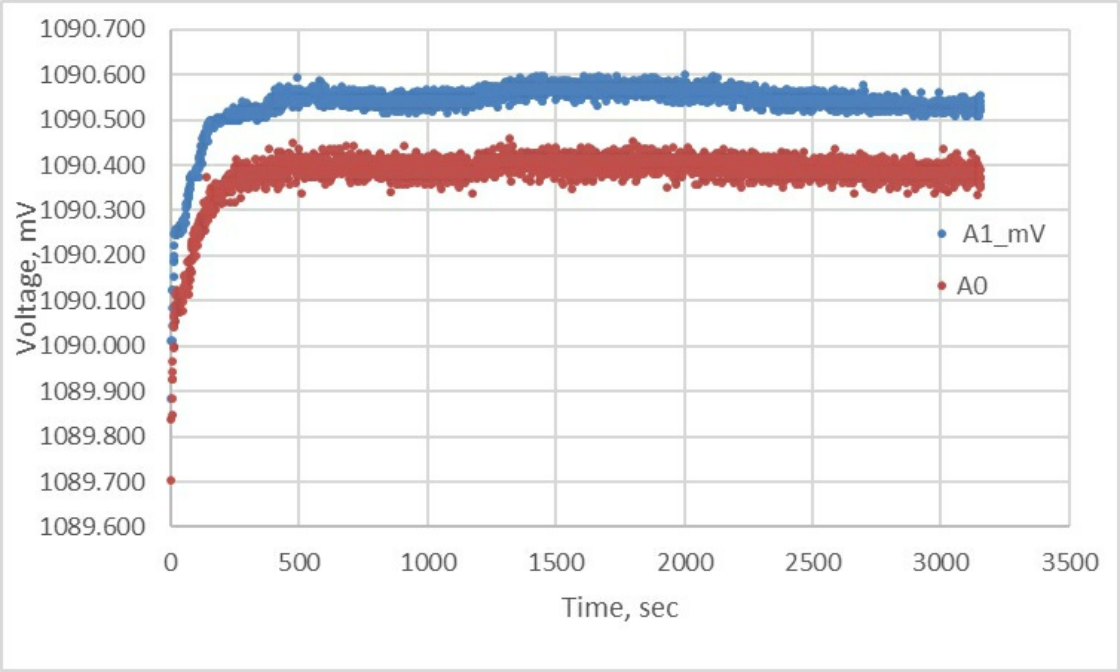


Figure 3.16 Measured voltage on the pot as a differential measurement (top trace) and as a single-ended measurement (bottom trace), immediately after the pot was turned. The scale is 100 uV/div.

This initial voltage change was only about 500 uV out of 1 V or 0.05%. It is probably a mechanical relaxation of the slider on the resistor pot. Once it stabilized, it was very stable, to within the 100 uV of peak to peak voltage noise on the pot.

Another confirmation of the importance of the mechanical stability of the resistor pot is illustrated when I tapped the table. This mechanical vibration jiggled the slider arm on the resistive ring inside the resistor pot, changing its position slightly, and changing the resistor voltage divider.

This reveals itself as small jumps in the voltage of the divider after tapping the table. This is shown in Figure 3.17.

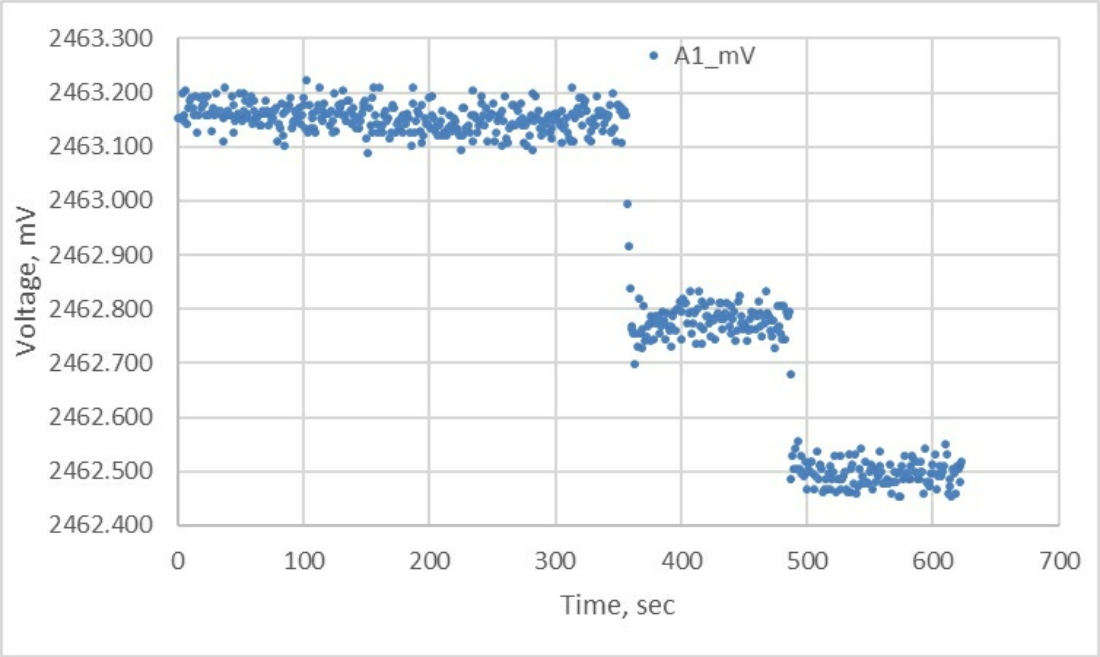


Figure 3.17 Measured single-ended voltage on the resistor pot after a few light taps of the table top. The scale is 0.1 mV/div. The steps are about 0.4 mV.

This suggests that we can use a resistor pot to adjust an offset voltage with a stability on the order of 100 uV. This is providing we keep the source mechanically stable.

3.8 Experiment: Resistor voltage divider

Sometimes, we will want to use a reduced voltage from another source using a fixed voltage divider with carbon resistors. If we use just a few simple carbon resistors, 10k each, in a 2 to 1 voltage divider, do we introduce any additional noise other than the source noise?

In a resistor, there will always be some Johnson, or thermal noise. This is ideally,

$$V_{Johnson} = \sqrt{4kT \cdot R \cdot BW} = 7.43[nV]\sqrt{R \cdot BW[Hz]}$$

This is the rms thermal noise in a resistor, with resistance of R Ohms, in a bandwidth, BW.

In our case, since we are measuring over a 1 second interval, the bandwidth is 1 Hz. For a 10k Ohms resistor, the Johnson noise expected is about

$$V_{Johnson} = 7.43[nV]\sqrt{10^4 \times 1} = 0.74uV$$

This is below the level we can measure with this set up. We would not expect the Johnson noise to add measurable noise to the resistor voltage divider using even 10k resistors.

However, carbon resistors have shown to have excess noise which is probably due to 1/f noise. We might expect some additional resistor noise above 0.74 uV.

To set up this experiment, we use the scale of 1x and a differential measurement using the local ground of the AD584 reference as the voltage source and the local ground of the low resistor as inputs to one of the differential inputs.

We used a pair of 10k carbon resistors in the voltage divider and fed it with the same 2.5 V output from the AD584. This configuration is shown in Figure 3.18.

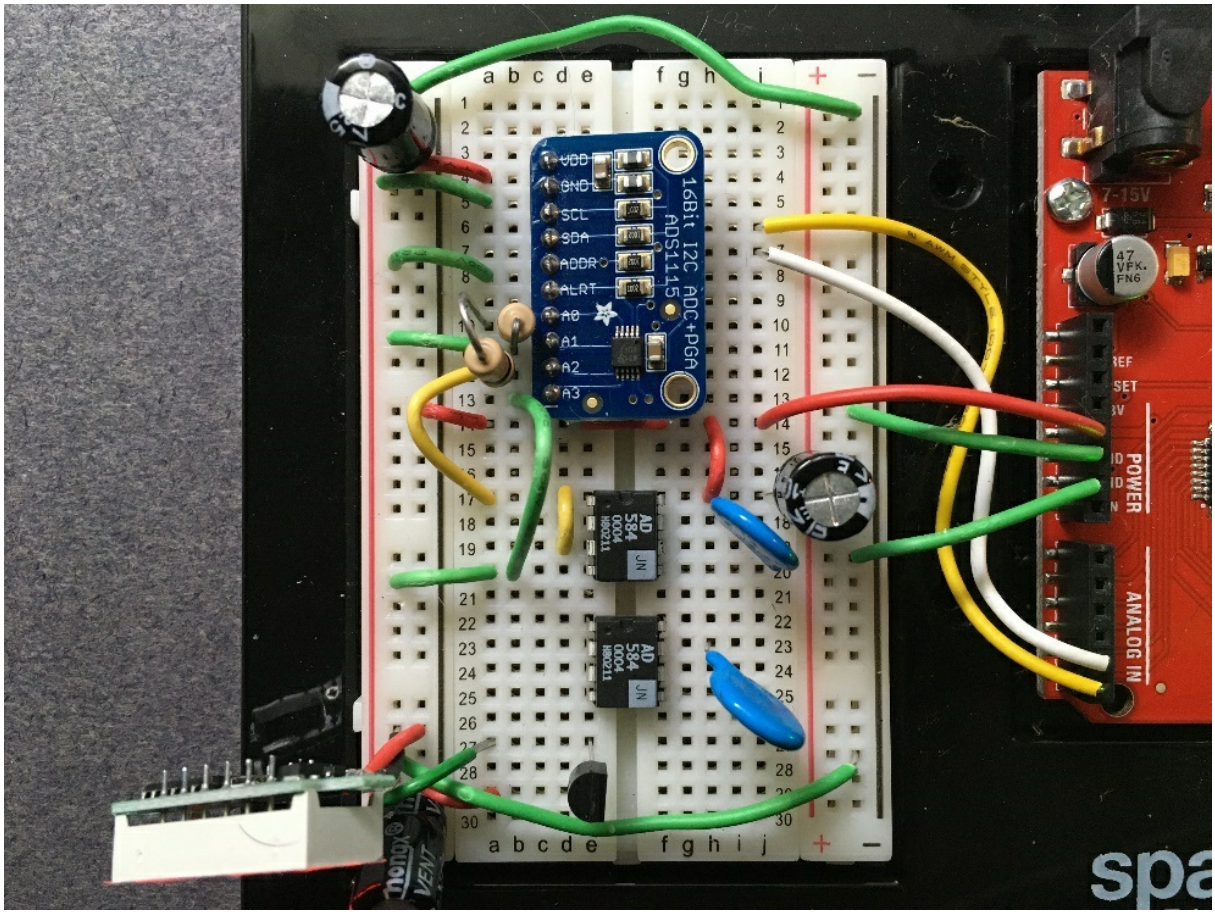


Figure 3.18 Measurement system to compare the AD584 voltage with a 2 to 1 voltage divider version using 2, 10k resistors and set up for differential measurements.

In both cases, we are measuring the differential voltage: the AD584 output and its local ground and the differential voltage across one of the 10k resistors. These measured voltages over a 30-minute period are shown in Figure 3.19.

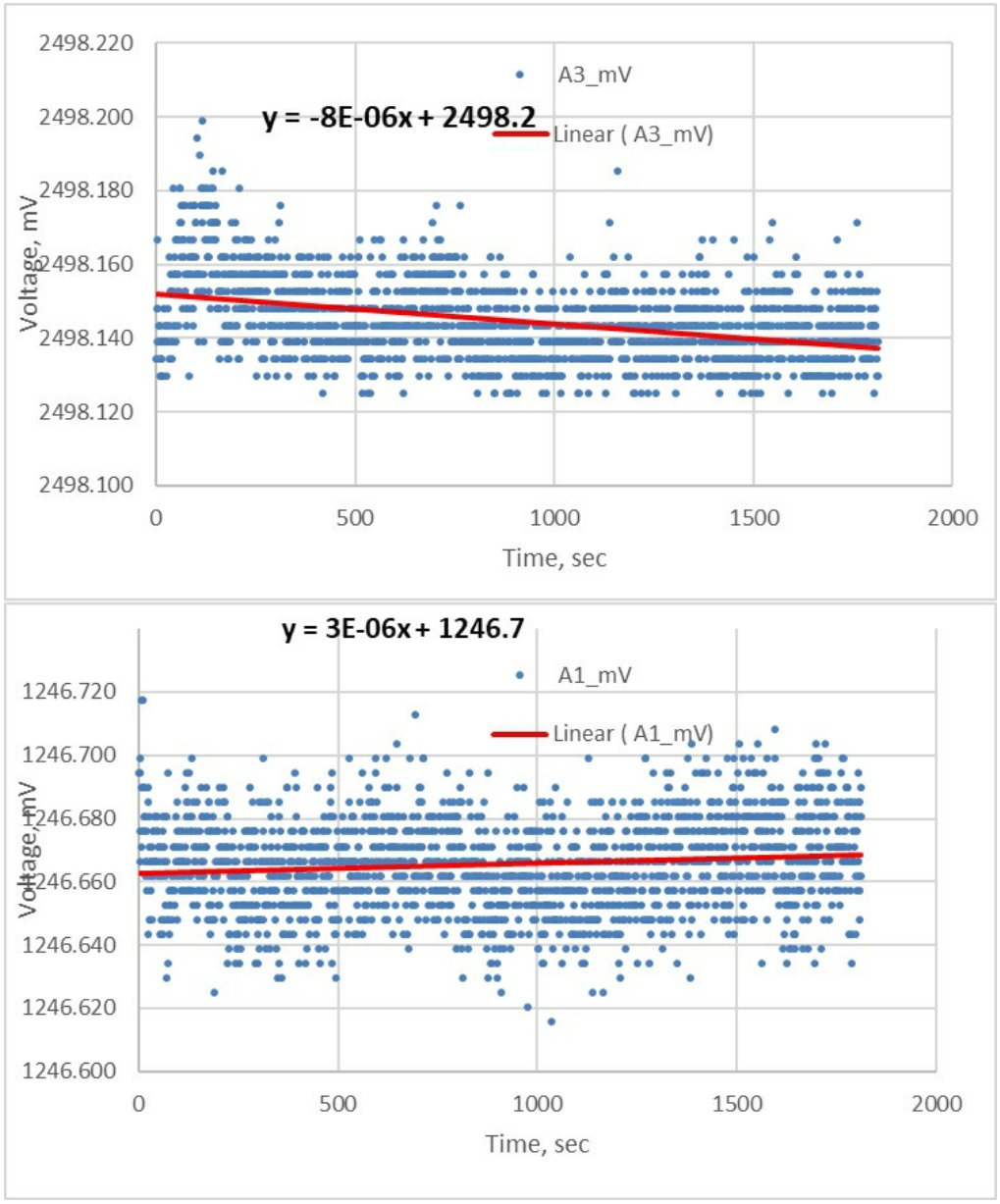


Figure 3.19 Measured noise on the AD584 reference source (top) and a 10k voltage divider (bottom), on the same scales of 20 uV/div. The noise in the voltage divider is twice the noise from the AD584 source.

The noise in the AD584 is about 11 uV, exactly as expected with the PGA set for 1x, using its local ground reference in a differential measurement. The noise in the 10k carbon voltage divider is about 15 uV. They are both at the noise limit of the ADS1115.

Even though the voltage across the resistor is ½ that of the AD584, we see the noise levels are comparable, around 12 uV rms. There may be a slightly higher rms noise on the 10k resistor, but it is at the noise floor of the ADS1115 and is an indication that the resistor divider is not adding any additional, measurable noise.

The resistance of a carbon resistor is temperature sensitive. Carbon has a negative temperature coefficient of -500 ppm/degC. This means that if the temperature increases, the resistance will decrease. If it is the lower resistor whose temperature increases, as when I touch it, its resistance should decrease and the voltage on this resistor should decrease as well.

As a simple test, I measured the voltage across the voltage divider and gently touched the lower resistor. I expected the temperature to increase and this to cause a decrease in resistance, which would result in a decrease in voltage across the resistor. This is exactly what was measured as shown in Figure 3.20.

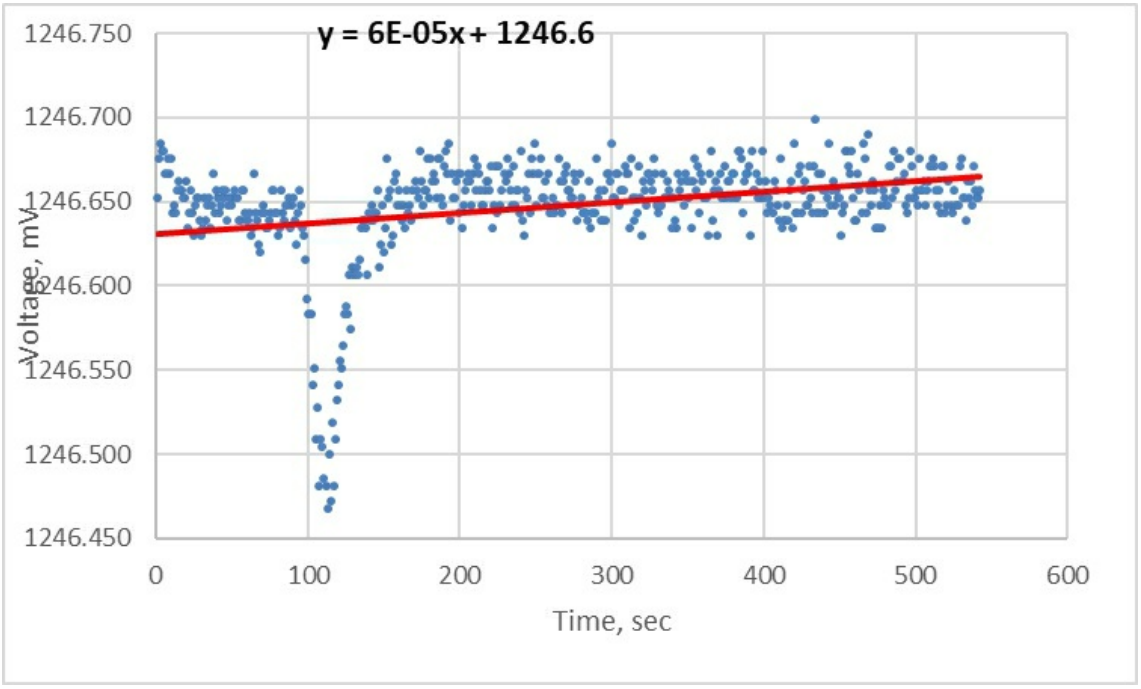


Figure 3.20 Measured voltage across the lower resistor in the voltage divider when I touched it. The

voltage dropped by 0.2 mV out of 1246 mV, or 0.016%.

We measured a change of 160 ppm. If the resistor's coefficient is the same as pure carbon, this would have been a fraction of a degree change in temperature.

3.9 Experiment: voltage across a 2.2 V Zener diode with 2 mA forward current

In this example, a very old Zener diode was used, in reverse bias mode, that was rated at 2.16 V. A 1k resistor connected to the 5 V supply. This means the reverse current was about 3 mA.

The reverse voltage on the Zener fluctuated about 600 uV out of about 2.2 V, over a period of about 15 minutes. This is still very stable, to within 0.03%, but easily measured with this system. Figure 3.21 shows the measured voltage on the Zener diode in this experiment.

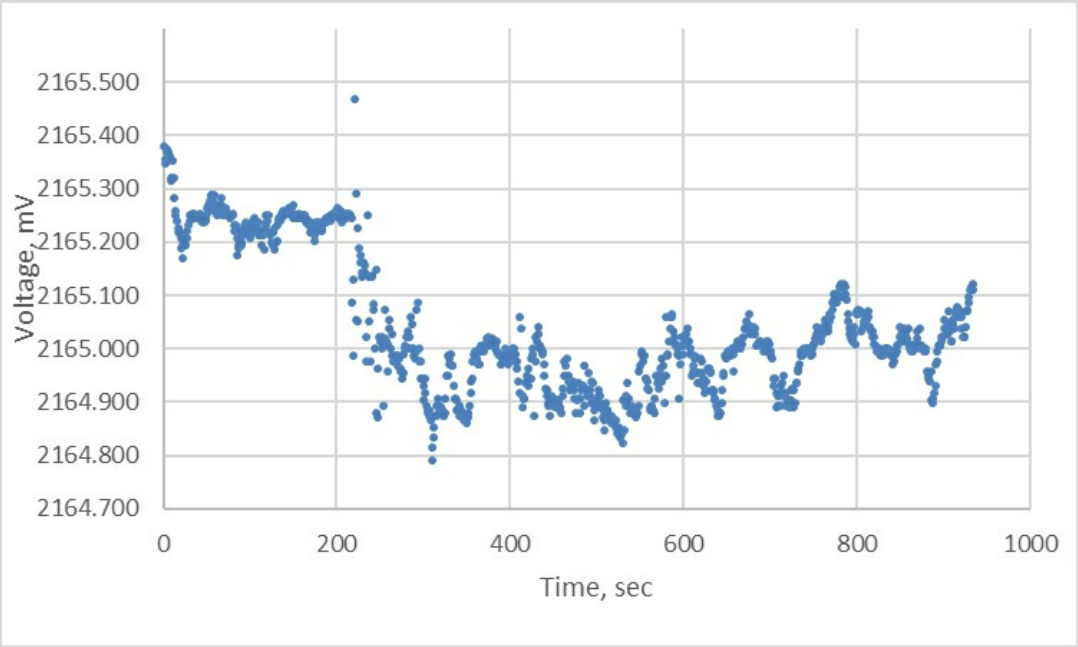


Figure 3.21 Measured voltage noise across a 2.1 V Zener biased with 3 mA of current.

The voltage range for this Zener was about 600 uV peak to peak, with a lot of short-term voltage fluctuations.

Again, not a suitable voltage reference source when very stable voltages are

required.

3.10 Experiment: Measuring the forward voltage drop of a silicon diode, 1N7000

Using the 10k resistor as a current limiting resistor and the 5 V rail as the voltage source, I measured the forward voltage drop across a 1N7000 silicon diode. It should be about 0.6 V. The measured voltage is shown in Figure 3.22.

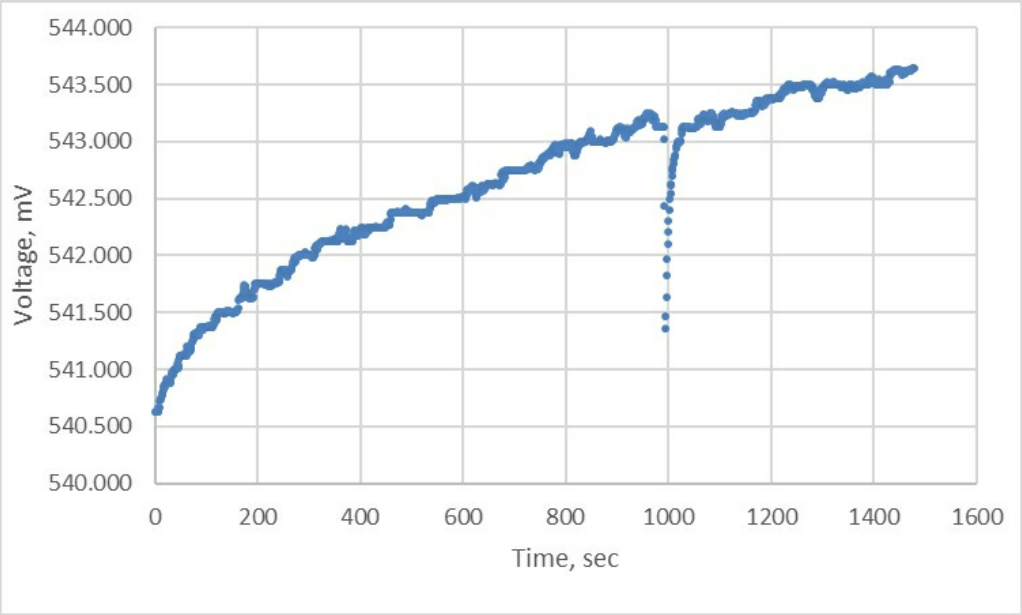


Figure 3.22 Measured forward voltage drop of a silicon diode. At 1000 sec, I slightly touched the diode. Its voltage dropped about 1.5 mV. The scale is 0.5 mV/div.

There was considerable drift in the measured voltage across the diode, on this scale of 0.5 mV/div. I suspected this was due to temperature changes in the diode, cooling off after I had handled it.

To test this condition, I touched the diode and slightly warmed it up. I expected the forward voltage to decrease with increasing temperature. In fact, the slight touch to the diode, at the 1000 second point, dropped the forward voltage drop about 1.5 mV, as shown above.

3.11 Experiment: forward voltage drop of an ultra-bright white

LED

Instead of a silicon diode, a white LED was used, with a 10k resistor to limit the current. The voltage across the 10k resistor is about 5 V – 2.58 V = 2.42 V. This means the current through the LED was about 2.42 V/10k = 0.242 mA, a very small amount of forward current.

After inserting the LED in the board, the voltage measured showed a long time to reach steady state. After 4 minutes, the forward voltage stabilized at about 2.579 V. This is shown in Figure 3.23.

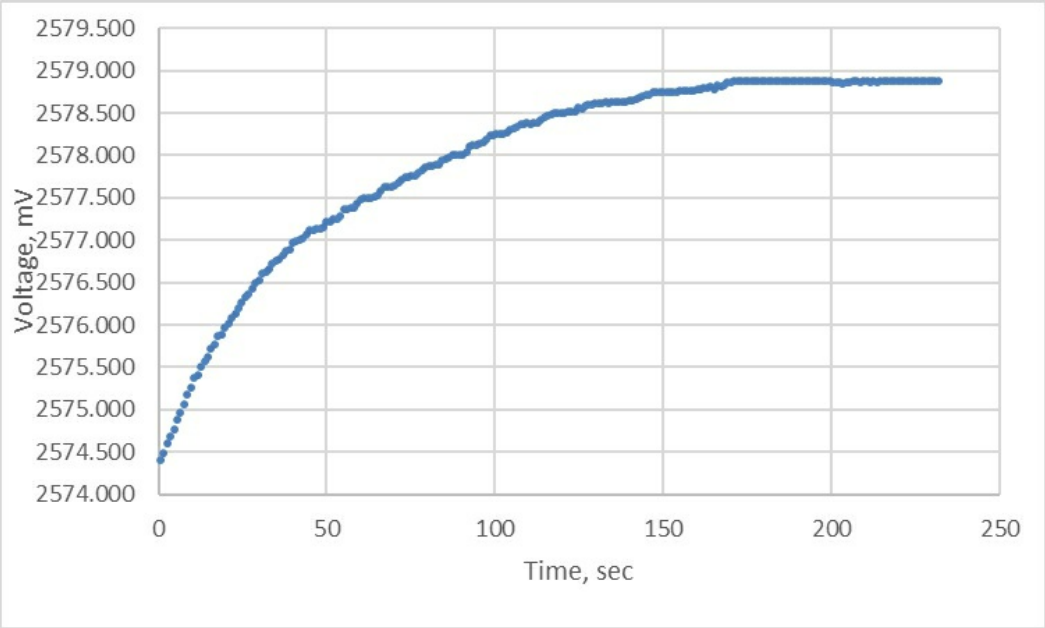


Figure 3.23 Measured forward voltage drop across the white LED after initially inserting into the solderless breadboard. The scale is 0.5 mV/div.

After the voltage had stabilized, it was measured for 45 minutes. The small-scale fluctuations are shown in Figure 3.24.

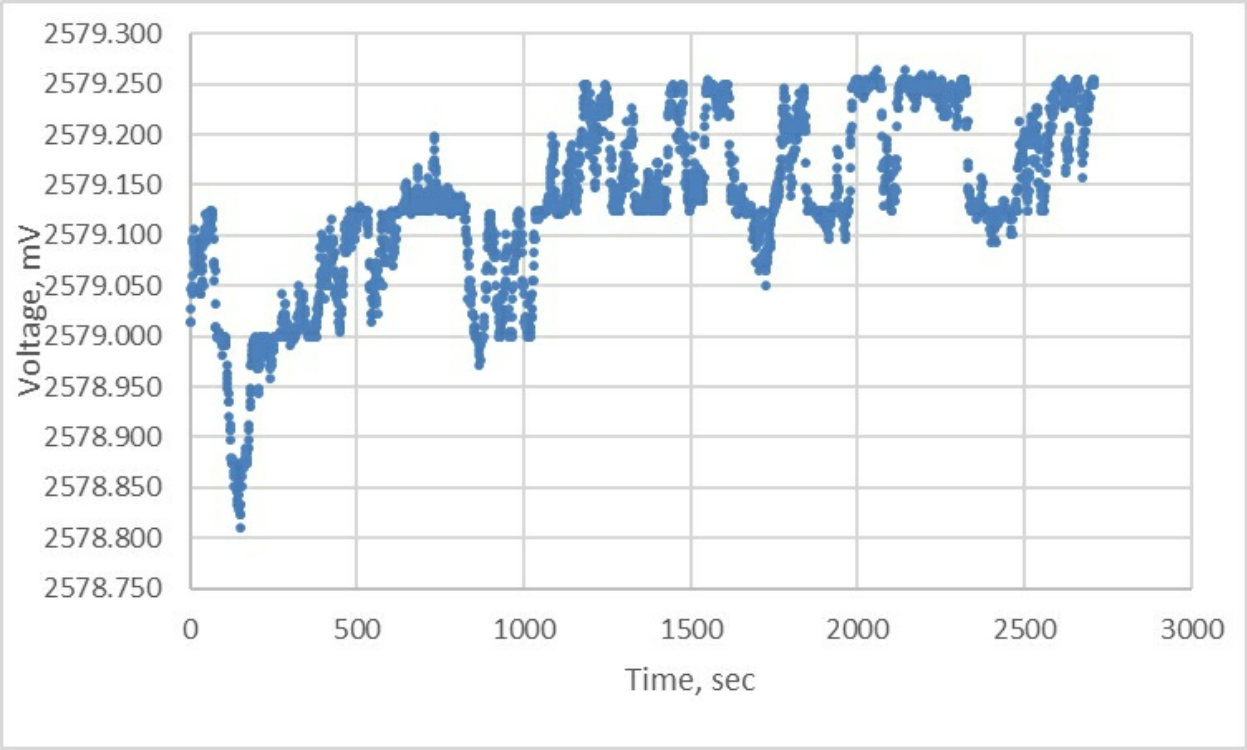


Figure 3.24 Forward voltage noise of a white LED with about 0.23 mA of forward current. The scale is 50 uV/div.

It is interesting that in this LED, there seem to be jumps in forward voltage on the order of 100 uV, out of the 2.579 forward voltage This is a change of 0.04%, very tiny, but very noticeable.

A second ultrabright red LED was measured with a 1k resistor. This measurement is shown in Figure 3.25.

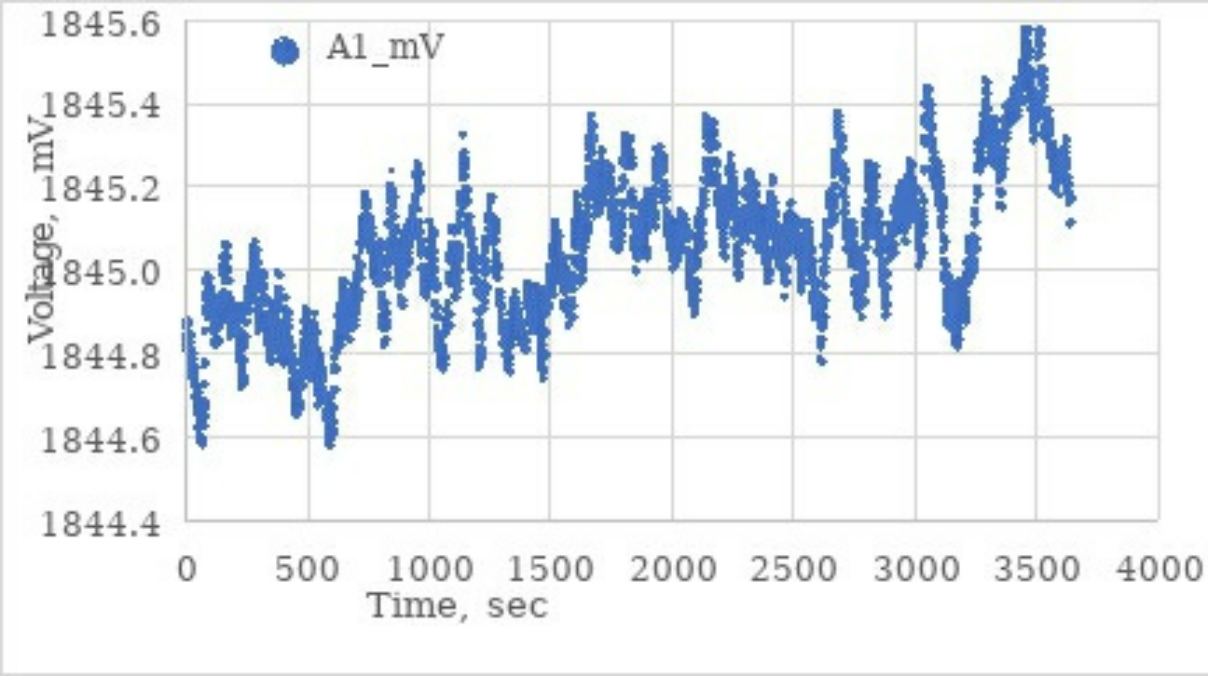


Figure 3.25 Measured forward voltage drop across a red LED with a 1k resistor.

3.12 A hint at temperature measurements

In addition to measuring the voltage stability of various voltage sources I also looked at the TMP36. Figure 3.26 shows a measurement over a 1-hour period of the TMP36 voltage level.

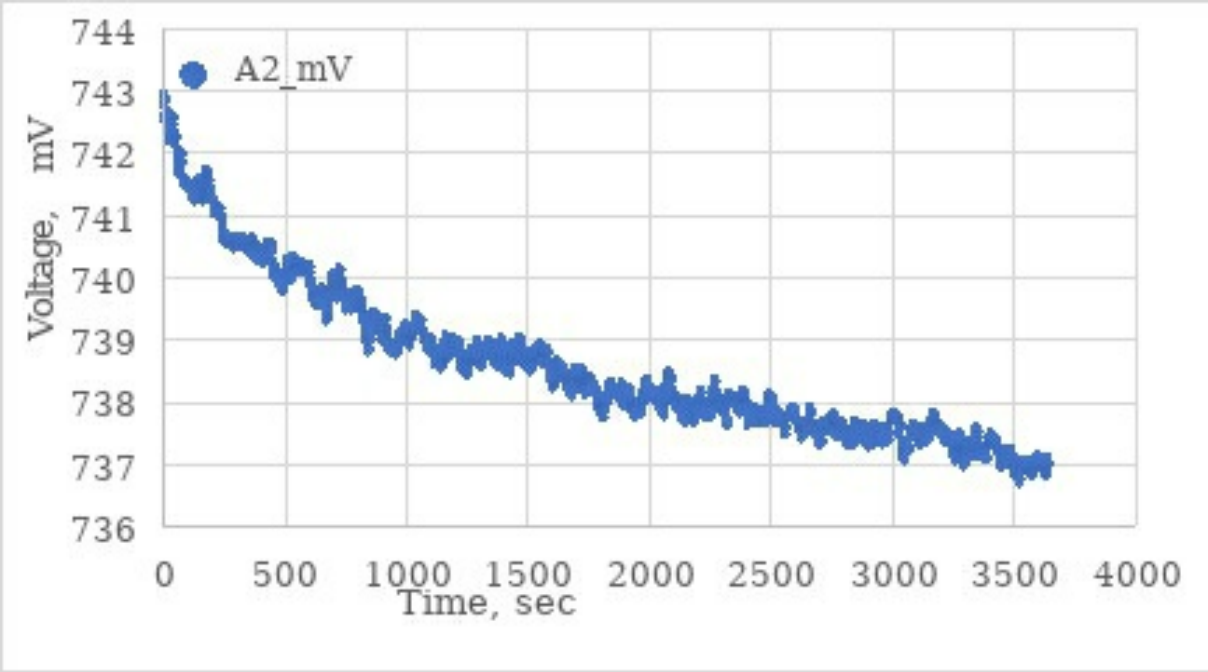


Figure 3.26 Measured voltage on the TMP36 sensor.

The scale is 1 mV/div. This is a temperature sensitivity of 0.1 degC/div. This shows the actual temperature drift in the ambient room over a 1-hour period.

The noise level is about 0.05 degC with slow and steady drift.

The voltage noise of the system is well below 100 uV. This is 0.01 degC. With confidence in the measurement system, we can now progress to using this system to measure very small temperature changes.

3.13 Summary of the lessons learned so far

1. *The noise floor of the ADS1115 is less than 1 LSB with averaging over the integration time. On the 1x gain scale, averaged over 1 second, this is 24 uV. On the 2x gain scale, this is 12 uV and on the 16x scale, this is 1.5 uV.*
2. *There is about 10 uV of rms noise on the ground lead of the ADS1115 IC. To avoid this noise in measurements, use a differential measurement of the voltage source between the source's positive and negative terminals.*
3. *If excess noise on the order of 10 uV is not important, single-*

ended voltage measurements are just fine. For the lowest noise measurements, always use differential measurements.

4. *The noise in two independent AD584 reference sources will have a typical peak to peak noise including drift of 10 uV.*
5. *The absolute accuracy of the AD584 2.5 V source is rated at 0.1% but may be much more accurate.*
6. *The ADS1115 internal band gap reference provides an absolute calibrated voltage accuracy with no additional effort, of better than 0.1% absolute accuracy.*
7. *When you need an external, stable reference voltage, use the AD584.*
8. *A battery will show some voltage drift due to its previous usage history and its cell voltage changing with the ambient temperature. This can vary by a few millivolts over an hour period. Don't use a battery as a long-term stable voltage source.*
9. *The USB hub 5 V source will show noise on the order of a few mV.*
10. *When using an Arduino Uno board, for the most stable 5 V rail, use an external 9-12 V external power source so that the 5 V LDO on the board supplies the 5 V rail.*
11. *For a variable voltage, use a resistor pot and the 5 V LDO rail or an AD584 reference source. Wait for mechanical transients to die down and be careful of small vibrations.*
12. *Diodes have voltage fluctuations on the order of 1 mV.*
13. *The best and cost effective voltage reference to use is an AD584 voltage reference.*

Chapter 4. The Complete Sketch for Data Acquisition and Analysis

The sketch I wrote will set up the ADS1115, record average number of points from selected channels and print them to an excel spreadsheet. To make it general, easy to modify and flexible to quickly make changes, it is written as functions using the tabs.

4.1 Setting up for typical measurements

Understanding that the rms noise level will decrease with the number of averages, and that the typical response time we care about is with a 1 second averaging time, we can set up a standard measurement system.

We can measure up to 4 channels of single-ended measurement or 2 channels of differential measurement, or any combination.

Each measurement can be on a different PGA scale setting.

We can measure alternating channels over a fixed averaging time. This would result in lower noise.

When we measure at a sampling rate of 108 SPS for a total time of 1 second, we will take a total of 108 measurements. If we measure 2 channels, there will be 108/2 = 54 measurements for each channel averaged into each measurement. If we measure 3 channels, there will be 108/3 = 36 samples averaged in each second. If we measure 4 channels, there will be 108/4 = 27 samples averaged in each channel.

There is very little additional time taken up in the overhead of changing the gain of the PGA or setting up the multiplexer for a specific channel connection. This means that each of the specific measurements can be any combination of scale and channels. The generic set up for each channel measurement is here:

```
    if (nChannels2meas >= 1) {
        ads.setGain(GAIN_SIXTEEN);
        //arrADS_meas_ADU[0] = ads.readADC_SingleEnded(0) * 1.0 + arrADS_meas_ADU[0];
        arrADS_meas_ADU[0] = ads.readADC_Differential_2_3() * 1.0 + arrADS_meas_ADU[0];
        arr_mV_per_ADU[0] = ads.voltsPerBit() * 1000.0F;
}
```

4.2 The experimental measurement system

We can use any Arduino with an I2C bus that understands the ADS1115 library. Since the Uno is so cheap and robust with just enough performance, all the experiments in this issue of the HackingPhysics Journal were performed with the low-end Arduino Uno. The specific version we used in this issue is the Sparkfun RedBoard. It comes with an adjacent solderless breadboard to conveniently set up the ADS1115.

In principle, it's not necessary to use a solderless breadboard to connect the ADS1115 to the Arduino. In fact, later in this issue, we show how to modify a USB cable to place the ADS1115 module close to the senor's analog voltage sources and the Arduino far away.

In this case, simple jumper wires can connect between the sensor and the ADS1115, with no solderless breadboard needed. But, for experimenting with the noise characterization and voltage stability experiments, it was much more convenient to use the solderless breadboard to mount the ADS1115 module and the various voltage sources.

The complete system to perform all the voltage experiments with one ADS1115 is shown in Figure 4.1.

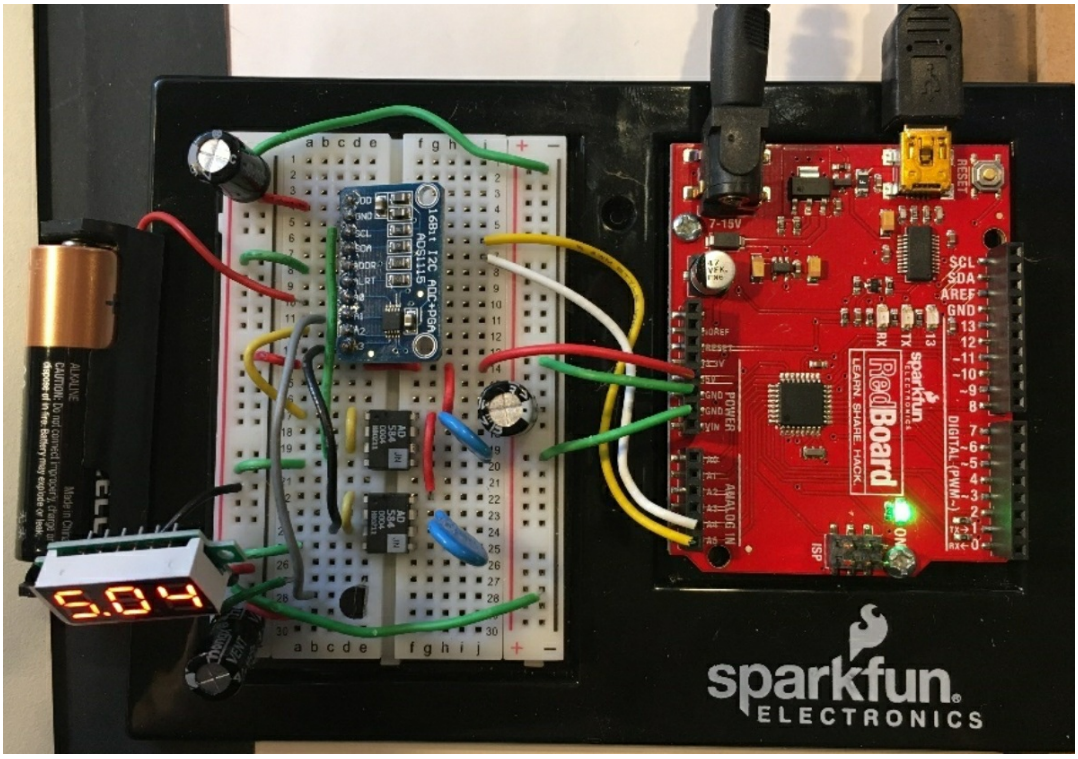


Figure 4.1 The complete system with a Sparkfun Redboard and ADS1115 module with two AD584 voltage reference chips, a battery as a reference and a TMP36 temperature sensor.

4.3 Overview of the sketch

A general-purpose sketch was written which was used for all the experiments in this issue of Hacking Physics Journal. It was partitioned into seven sections, created as functions:

- *Main section*
- *Special function to calc the rms values*
- *Measure 4 ADC channels interleaved*
- *Print the data in degC to excel*
- *Print the data in degF to excel*
- *Print the data in mV to excel*
- *Record data like a scope and print to excel*

Each section was used in a separate tab to make the sketch easier to navigate. Each function was called as needed in the main sketch.

To recreate this sketch, you can copy and paste from this eBook version, directly into a blank sketch using the tabs or place all the functions after each other, after the void loop() function.

Alternatively, the complete sketch has been posted here and can be downloaded by clicking this link. Unzip this file and the sketch will open by double clicking the elbADS4channels file.

Each section is detailed below.

4.4 The main section

This section is the master controller. All the variables are defined and set up, and the specific functions are called, as needed.

Here is the complete main section:

```
// www.HackingPhysics.com data acquisition
// Eric Bogatin
#include <Wire.h>
#include <Adafruit_ADS1015.h>

//ADS1115 specific- always read all nChannels channels as SE or diff
Adafruit_ADS1115 ads;

int nChannels2meas = 4;
int iFlag_XL = 1; //1 means print to excel, 0 means do not
int pinScope = 3; // SELECTS THE pin for scope measurements
long npts_run = 3600;// total number of pts to record into excel

float ADS_meas_ADU;
float ADS_meas_ADU_ave;
float ADS_meas_ADU_sigma;
float arrADS_meas_ADU[4];
float arrADS_meas_mV[4];  // has the measured values on each channel
float arrADS_meas_degC[4];
float arrADS_meas_degF[4];
float arr_mV_per_ADU[4];
float V_meas_ADU;
float V_meas_mV;
float V_meas_degC;
float V_meas_degF;

//excel specific variables:

int iFlag_XL_labels = 1;// will print labels
long iTime_SaveInterval_min = 10; // how often to save strip chart
long iTime_LastSave_msec = 0;

//scope:

int npts_scope_ave = 1;
const int npts_scope_buffer = 300;// max 301 at 16-bit
float arrVscope_ADU[npts_scope_buffer + 1]; // need to make float for calculations
int delay_XL_msec = 150;// used for printing 500 scope values

float Time_point_sec;
float Time_X_usec;
float Time_point_msec;
long iTime0_usec;
long index_val = 0;

//stripchart recorder:
long iTime_ave_usec = 1000;

//temp calibration
float arrM_degC_per_mV[5];
float arrT0_degC[5];

float T1_degC;
float V1_0_mV;
float V1_1_mV;
float V1_2_mV;
```

```
float V1_3_mV;

///*** if one temperature point, use T2
float T2_degC = 24.0;
float V2_0_mV = 744.36;
float V2_1_mV = 735.56;
float V2_2_mV = 742.2;
float V2_3_mV = 739.3;


//int nPLC = 0; //set to 0 for hard coded averaging
//int nPLC = 1; //17 msec/point
//int nPLC = 3; //50 msec/point
//int nPLC = 6; //100 msec/point
//int nPLC = 12; //200 msec/point
//int nPLC = 30; //500 msec/point
int nPLC = 60; //1sec/point
//int nPLC = 60*5; //5sec/point
long iTime_stop_usec;
long iCounter1;
long iTime_start_usec;

void setup() {
    Serial.begin(2000000);
    if (nPLC != 0) {
        iTime_ave_usec = (nPLC / 60.0) * 1.0e6;
  }
    ads.begin();
    ads.setSPS(ADS1115_DR_128SPS);// multiples of 8, 128 is default
    iTime0_usec = micros();

    for (int i = 0; i < nChannels2meas; i++) {
        arrM_degC_per_mV[i] = 0.1;
        arrT0_degC[i] = -50.0;
  }

    //******m=(T2-T1)/(V2-V1)
    //arrM_degC_per_mV[0]= 0.1;   //(T1_degC-T2_degC)/(497.7-941.5);
    //arrM_degC_per_mV[1]= 0.1;    //(T1_degC-T2_degC)/(499.0-944.5);
    //arrM_degC_per_mV[2]= 0.1;  //(T1_degC-T2_degC)/(499.3-944.0);
    //arrM_degC_per_mV[3]= 0.1;   //(T1_degC-T2_degC)/(494.0-939.0);

    //*****T0 = T2 - m x V2
    //arrT0_degC[0] = T2_degC - arrM_degC_per_mV[0] * V2_0_mV;
    //arrT0_degC[1] = T2_degC - arrM_degC_per_mV[1] * V2_1_mV;
    //arrT0_degC[2] = T2_degC - arrM_degC_per_mV[2] * V2_2_mV;
    //arrT0_degC[3] = T2_degC - arrM_degC_per_mV[3] * V2_3_mV;

}
void loop() {
    index_val++;

  func_meas_ADC();
    //func_print_mV();
  func_print_degF();
    //func_print_degC();
```

```
    //func_calcRMS();
    //func_scope();

    if (index_val >= npts_run) {
        if (iFlag_XL == 1) {
            Serial.println("SAVEWORKBOOK");
            Serial.println("BEEP");
            Serial.println("BEEP");
            Serial.println("BEEP");
            Serial.println("BEEP");
     }
        delay(1e9);
   }
}
```

4.5 Calculate rms values

As a special case, this function will calculate the rms value for a collection of measured points, averaged for some period of time. This is really to confirm that the rms value will decrease with the square root of the number of measurements.

```
// calc the rms noise with npts_ave

void func_calcRMS ( ) {
    ads.setGain ( GAIN_SIXTEEN ) ;
    arr_mV_per_ADU [ 0 ] = ads . voltsPerBit ( ) * 1000.0F ; // Sets the millivolts per bit

    for ( npts_scope_ave = 1 ; npts_scope_ave <= 500 ; npts_scope_ave ++ ) {
        ADS_meas_ADU_ave = 0.0 ;
        ADS_meas_ADU_sigma = 0.0 ;

        for ( int j = 1 ; j <= npts_scope_buffer ; j ++ ) {
            ADS_meas_ADU = 0.0 ;
            iCounter1 = 0 ;
            for ( int i = 1 ; i <= npts_scope_ave ; i ++ ) {
                // ADS_meas_ADU = ads.readADC_SingleEnded (pinScope) * 1.0 + ADS_meas_ADU;
                ADS_meas_ADU = ads . readADC_Differential_2_3 ( ) * 1.0 + ADS_meas_ADU ;
                iCounter1 ++ ;
     }
            arrVscope_ADU [ j ] = ADS_meas_ADU / ( iCounter1 * 1.0 ) ;
            ADS_meas_ADU_ave = ADS_meas_ADU_ave + arrVscope_ADU [ j ] ;
    }
        ADS_meas_ADU_ave = ADS_meas_ADU_ave / ( npts_scope_buffer * 1.0 ) ;
        for ( int i = 1 ; i <= npts_scope_buffer; i ++ ) {
            ADS_meas_ADU_sigma = ADS_meas_ADU_sigma + pow ( ( arrVscope_ADU [ i ] - ADS_meas_ADU_ave ) , 2 ) ;
    }
        ADS_meas_ADU_sigma = sqrt ( ADS_meas_ADU_sigma / ( npts_scope_buffer * 1.0 ) ) ;
        Serial . print ( npts_scope_ave ) ; Serial . print ( "," );
        Serial . print ( ( ADS_meas_ADU_sigma * arr_mV_per_ADU [ 0 ] ) , 5 ) ; Serial . print ( "," ) ;
        Serial . print ( ( ADS_meas_ADU_ave * arr_mV_per_ADU [ 0 ] ) , 5 ) ; Serial . print ( "," ) ;
        Serial . println ( ) ;
  }
    delay ( 1e9 ) ;
}
```

This snippet of code can be added to a new tab or to the bottom of the main sketch. It is called in the main section by its function name.

4.6 Measure with the ADS1115

The main work horse section of code is the function that measures the data from the ADS1115 module.

This function will measure each of the four channels, as they are set up, interleaved, and average them over the time period. The four different numbers, plus the time of the measurement and the value of each channel converted from mV into degC or degF will be returned in an array.

A new measurement of the four channels will be returned each time this function is called.

In all cases, I use global variables that are defined in the main sketch. This way a change to one of the variables made inside a function will propagate to all the functions.

Here is the sketch:

```
/ returns V_meas_mV[i] of 4 channels
void func_meas_ADC() {
      // ads.setGain(GAIN_TWOTHIRDS); // 2/3x gain +/- 6.144V 1 bit = 3mV 0.1875mV (default)
      // ads.setGain(GAIN_ONE); // 1x gain +/- 4.096V 1 bit = 2mV 0.125mV
      // ads.setGain(GAIN_TWO); // 2x gain +/- 2.048V 1 bit = 1mV 0.0625mV
      // ads.setGain(GAIN_FOUR); // 4x gain +/- 1.024V 1 bit = 0.5mV 0.03125mV
      // ads.setGain(GAIN_EIGHT); // 8x gain +/- 0.512V 1 bit = 0.25mV 0.015625mV
      // ads.setGain(GAIN_SIXTEEN); // 16x gain +/- 0.256V 1 bit = 0.125mV 0.0078125mV

      for (int i = 0; i <= 3; i++) {
            arrADS_meas_ADU[i] = 0.0;
   }
      iCounter1 = 0;
      iTime_stop_usec = micros() + iTime_ave_usec;
      // start averaging loop
      while (micros() < iTime_stop_usec) {

            if (nChannels2meas >= 1) {
                  ads.setGain(GAIN_FOUR);
                  arrADS_meas_ADU[0] = ads.readADC_SingleEnded(0) * 1.0 + arrADS_meas_ADU[0];
                  //arrADS_meas_ADU[0] = ads.readADC_Differential_2_3() * 1.0 + arrADS_meas_ADU[0];
                  arr_mV_per_ADU[0] = ads.voltsPerBit() * 1000.0F; // Sets the millivolts per bit
      }

            if (nChannels2meas >= 2) {
                  ads.setGain(GAIN_FOUR);
                  arrADS_meas_ADU[1] = ads.readADC_SingleEnded(1) * 1.0 + arrADS_meas_ADU[1];
                  //arrADS_meas_ADU[1] = ads.readADC_Differential_0_1() * 1.0 + arrADS_meas_ADU[1];
                  arr_mV_per_ADU[1] = ads.voltsPerBit() * 1000.0F; // Sets the millivolts per bit
      }

            if (nChannels2meas >= 3) {
                  ads.setGain(GAIN_FOUR);
                  arrADS_meas_ADU[2] = ads.readADC_SingleEnded(2) * 1.0 + arrADS_meas_ADU[2];
                  //arrADS_meas_ADU[2] = ads.readADC_Differential_2_3() * 1.0 + arrADS_meas_ADU[2];
                  arr_mV_per_ADU[2] = ads.voltsPerBit() * 1000.0F; // Sets the millivolts per bit
      }

            if (nChannels2meas >= 4) {
                  ads.setGain(GAIN_FOUR);
                  arrADS_meas_ADU[3] = ads.readADC_SingleEnded(3) * 1.0 + arrADS_meas_ADU[3];
                  //arrADS_meas_ADU[3] = ads.readADC_Differential_2_3() * 1.0 + arrADS_meas_ADU[3];
                  arr_mV_per_ADU[3] = ads.voltsPerBit() * 1000.0F; // Sets the millivolts per bit
      }
            iCounter1++;
   }
```

```
    Time_point_sec = ((micros() - 0.5 * iTime_ave_usec)) * 1.0e-6;
    Time_point_msec = Time_point_sec * 1.0e3;

    for (int i = 0; i < nChannels2meas; i++) {
        arrADS_meas_ADU[i] = arrADS_meas_ADU[i] / (iCounter1 * 1.0);
        arrADS_meas_mV[i] = arrADS_meas_ADU[i] * arr_mV_per_ADU[i];
        arrADS_meas_degC[i] = arrADS_meas_mV[i] * arrM_degC_per_mV[i] + arrT0_degC[i];
        arrADS_meas_degF[i] = arrADS_meas_degC[i] * 1.8 + 32.0;
  }
}
```

To use this function, you first have to manually set up what gain and what sort of measurement you want for each channel, whether single-ended or differential.

Once it is set up, it is called in the main function in the loop. A counter in the loop decides how many measurements are taken.

4.7 Printing to excel

After each measurement is taken, the next step is to print the values into excel. I created a different function to print the value in mV, in degC or degF.

Which value to print is selected by the function that is called.

In excel, I am running the PLX-DAQ v2 macro. This is described in detail in: Science Experiments with Arduinos and HackingPhysics Journal vol 1 no 1, Jan 2020.

Here is the function to print the value of each channel in mV:

```
// print all 4 mV to the serial printer

void func_print_mV() {

    if (iFlag_XL == 1) {
        if (iFlag_XL_labels == 1) {
            //Serial.println("CLEARSHEET"); // clears entire sheet
            // will clear only some cells so we can keep formulas on sheet
            Serial.println("CLEARRANGE,A,1,G,20000");
            // clears from cell A1 to cell C3001
            //Serial.println("LABEL, index, averages, Time(sec), A0_degC, A1_degC, A2_degC, A3_degC");
            Serial.println("LABEL, index, averages, Time(sec), A0_mV, A1_mV, A2_mV, A3_mV");
            iFlag_XL_labels = 0;
    }
        // print to excel options:
        Serial.print("DATA,");
        Serial.print(index_val); Serial.print(", ");
        Serial.print(iCounter1); Serial.print(", ");
        Serial.print(Time_point_sec, 5); Serial.print(", ");
  }

    for (int i = 0; i < nChannels2meas; i++) {
        // Serial.print(arrADS_meas_degC[i], 4); Serial.print(", ");
        Serial.print(arrADS_meas_mV[i], 4); Serial.print(", ");
  }
    Serial.println();

    //save worksheet routine
    if (millis() > iTime_LastSave_msec + iTime_SaveInterval_min * 60000) {
        Serial.println("BEEP");
        Serial.println("SAVEWORKBOOK");
```

```
        iTime_LastSave_msec = millis();
    }
}
```

Just a few lines are changed to print the numbers as degC:

```
// print all 4 degC to the serial printer

void func_print_degC() {

    if (iFlag_XL == 1) {
        if (iFlag_XL_labels == 1) {
            //Serial.println("CLEARSHEET"); // clears entire sheet
            // will clear only some cells so we can keep formulas on sheet
            Serial.println("CLEARRANGE,A,1,G,20000");
            // clears from cell A1 to cell C3001
            //Serial.println("LABEL, index, averages, Time(sec), A0_degC, A1_degC, A2_degC, A3_degC");
            Serial.println("LABEL, index, averages, Time(sec), A0_degC, A1_degC, A2_degC, A3_degC");
            iFlag_XL_labels = 0;
    }
        // print to excel options:
        Serial.print("DATA,");
        Serial.print(index_val); Serial.print(", ");
        Serial.print(iCounter1); Serial.print(", ");
        Serial.print(Time_point_sec, 5); Serial.print(", ");
    }

    for (int i = 0; i < nChannels2meas; i++) {
        // Serial.print(arrADS_meas_degC[i], 4); Serial.print(", ");
        Serial.print(arrADS_meas_degC[i], 4); Serial.print(", ");
    }
    Serial.println();

    //save worksheet routine
    if (millis() > iTime_LastSave_msec + iTime_SaveInterval_min * 60000) {
        Serial.println("BEEP");
        Serial.println("SAVEWORKBOOK");
        iTime_LastSave_msec = millis();
    }
}
```

The third option is to print the data as degF.

```
// print all 4 degF to the serial printer

void func_print_degF() {

    if (iFlag_XL == 1) {
        if (iFlag_XL_labels == 1) {
            //Serial.println("CLEARSHEET"); // clears entire sheet
            // will clear only some cells so we can keep formulas on sheet
            Serial.println("CLEARRANGE,A,1,G,20000");
            // clears from cell A1 to cell C3001
            //Serial.println("LABEL, index, averages, Time(sec), A0_degC, A1_degC, A2_degC, A3_degC");
            Serial.println("LABEL, index, averages, Time(sec), A0_degF, A1_degF, A2_degF, A3_degF");
            iFlag_XL_labels = 0;
    }
        // print to excel options:
        Serial.print("DATA,");
        Serial.print(index_val); Serial.print(", ");
        Serial.print(iCounter1); Serial.print(", ");
        Serial.print(Time_point_sec, 5); Serial.print(", ");
    }

    for (int i = 0; i < nChannels2meas; i++) {
        // Serial.print(arrADS_meas_degC[i], 4); Serial.print(", ");
        Serial.print(arrADS_meas_degF[i], 4); Serial.print(", ");
    }
    Serial.println();

    //save worksheet routine
    if (millis() > iTime_LastSave_msec + iTime_SaveInterval_min * 60000) {
        Serial.println("BEEP");
        Serial.println("SAVEWORKBOOK");
        iTime_LastSave_msec = millis();
    }
}
```

4.8 Taking data fast like a scope

The last function will take data as quickly as possible, at 108 SPS, and write them into an array. The values in the array will be converted and then printed more slowly into excel.

```
// record measurements into a buffer and slowly print out the buffer

void func_scope() {

    if (iFlag_XL == 1) {
        // if we print to excel, set it up
        Serial.println("CLEARRANGE,A,1,C,9001");
        Serial.println("LABEL, index, Time(sec), mV");
  }

    // initialize array
    for (int i = 1; i <= npts_scope_buffer; i++) {
        arrVscope_ADU[i] = 0.0;
  }
    ads.setGain(GAIN_SIXTEEN);

    //begin quickly filling int array with averaged points
    iTime_start_usec = micros();
    for (int i = 1; i <= npts_scope_buffer; i++) {
        for (int j = 1; j <= npts_scope_ave; j++) {
            //arrVscope_ADU[i] = arrVscope_ADU[i] + ads.readADC_SingleEnded(pinADS1115);
            //arrVscope_ADU[i] = arrVscope_ADU[i] + ads.readADC_Differential_0_1();
            arrVscope_ADU[i] = arrVscope_ADU[i] + ads.readADC_Differential_2_3();
    }
        arrVscope_ADU[i] = arrVscope_ADU[i] / (npts_scope_ave * 1.0);
  }
    Time_X_usec = (micros() - iTime_start_usec) * 1.0;
    Time_X_usec = Time_X_usec / (npts_scope_ave * npts_scope_buffer * 1.0);

    // begin printing out formated array contents
    for (int i = 1; i <= npts_scope_buffer; i++) {
        V_meas_ADU = arrVscope_ADU[i];
        V_meas_mV = V_meas_ADU * ads.voltsPerBit() * 1000.0F;;
        V_meas_degC = V_meas_mV / 10.0 - 50.0;
        V_meas_degF = V_meas_degC * 1.8 + 32.0;
        Time_point_msec = i * Time_X_usec * 1.0e-3;
        Time_point_sec = i * Time_X_usec * 1.0e-6;
        if (iFlag_XL == 1) {
            // if printing to excel, format the data
            Serial.print("DATA,");
            Serial.print(i); Serial.print(", ");
            Serial.print(Time_point_sec, 5); Serial.print(", ");
            //Serial.println(Temp_degF, 3);
            Serial.println(V_meas_mV, 4);
            delay(delay_XL_msec);
    }
        else {
            //if not printing to excel, format for serial monitor
            Serial.print(i); Serial.print(", ");
            //Serial.print(Time_X_usec); Serial.print(", ");
            Serial.print(Time_point_msec, 3); Serial.print(", ");
            //Serial.print(100); Serial.print(", ");
            //Serial.print(V_meas_ADU, 3); Serial.print(", ");
            Serial.println(V_meas_mV, 4);
            //Serial.println(V_meas_degC);
            //Serial.println(V_meas_degF);
    }
  }
    // done exporting the array
    if (iFlag_XL == 1) {
        Serial.println("SAVEWORKBOOK");
        while (1 != 2) {
            Serial.println("BEEP");
            delay(1000);
    }
  }
    delay(2000000); // wait to view screen
}
```

The slower printing speed into excel is to prevent excel from crashing.

Chapter 5. Building a Simple Four Channel Temperature Measurement System

With this precision voltage measurement system with measurements printed directly into excel every second, it is the perfect system to measure small temperatures using a TMP36 temperature sensor.

This sensor is sensitive to 10 mV/degC, or 0.1 degC/mV. If we can see voltage changes of 10 uV or 0.01 mV, we can see temperature changes of 0.1 degC/mV x 0.01 mV = 1 milliDegC. This is a very small change.

5.1 The TMP36 temperature sensor

The temperature sensor we will use in all these experiments is the TMP36 temperature sensor. An example is shown in Figure 5.1. This is the simplest sensor to use and a very useful one.

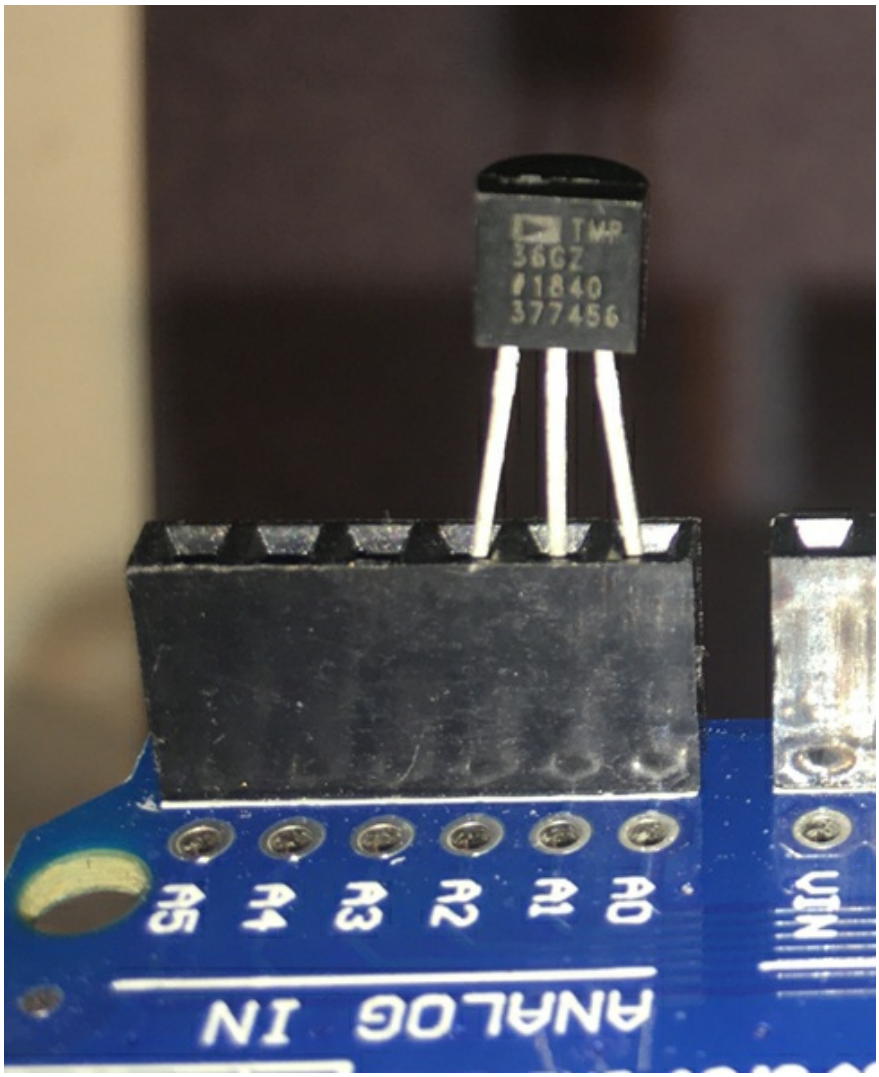


Figure 5.1 A TMP36 temperature sensor plugged directly into the header pins of an Arduino.

Generally, these can be purchased from Sparkfun for $1.50 and similar price on Amazon or AliExpress.

The TMP36 measures temperature based on the property that the current through a diode depends a little on temperature. You can learn a little more about how it works as well as all the other important information about how to use this sensor from its datasheet.

Figure 5.2 shows the TMP36 with its pin assignment. This is the view of the TMP36 held with the pins facing you and the flat side up. Pin 1 on the left is + 5 V, pin 2 is the output voltage and pin 3 on the right is the ground reference connection.

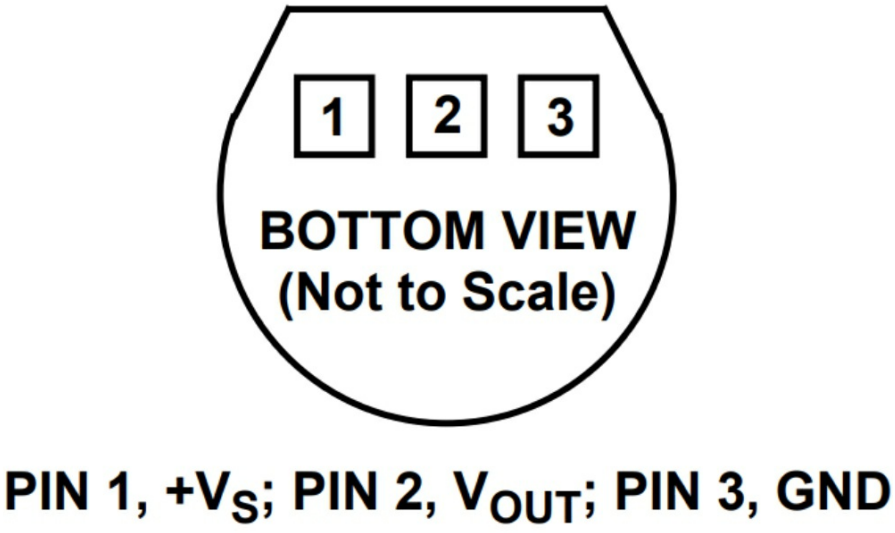


Figure 5.2. Pinout for the TMP36 temperature sensor, as viewed from the bottom with the legs pointing at you.

There are three pins to the sensor. To operate, we apply 5 V to pin 1, 0 V to pin 3 and measure the voltage on pin 2, relative to the ground pin. The circuit inside this sensor will convert its temperature into the voltage on pin 2. The sensor voltage is measured between pins 2 and 3. The output voltage is linearly dependent on the temperature.

The graph in Figure 5.3 shows how the voltage we measure between the middle pin and the ground pin is related to the temperature of the sensor. We call this sort of curve, a calibration curve.

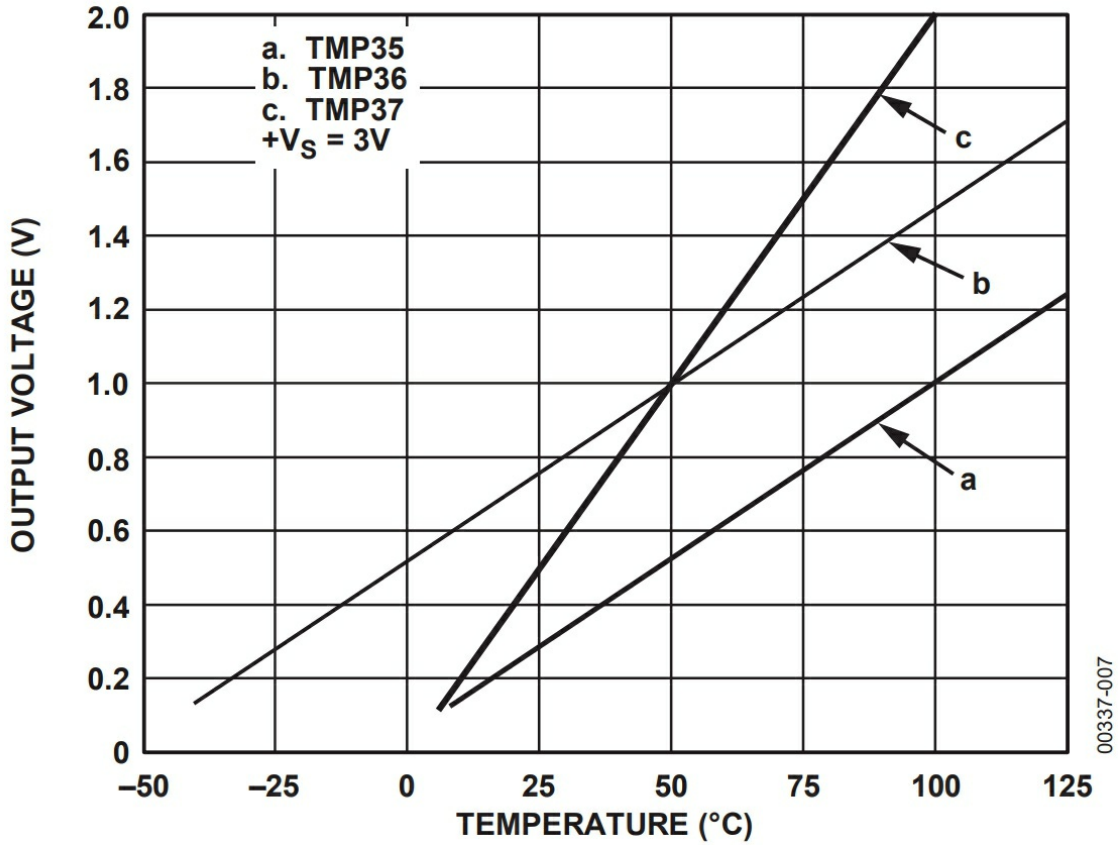


Figure 5.3. The calibration curve for the TMP36. The curve labeled b is the one to use.

The sensor is designed with a linear map between the voltage measured and the temperature. This means the connection can be described by a linear equation, in the form of:

T[degC] = m x V[mV] + T0[degC]

The slope term, m, is the sensitivity which is 0.1 degC/mV. The intercept, T0, is the temperature when the voltage is 0 mV, which is -50 degC. This conversion is:

T[degC] = 0.1 degC/mV x V[mV] – 50 degC

and, of course, the conversion from degC to deg F is

T[degF] = T[degC] x 9/5 + 32 degF

5.2 Configuring the TMP36 sensor

The TMP36 sensor is a 3-terminal sensor in a TO-92 plastic package. The simplest way of interfacing it to the ADS1115 is by using three wires of a ribbon cable soldered to the TO-92 connector on one end to a 3-pin header on the other end.

The ribbon cable keeps the wires close together and easy to manipulate. Keeping the signal out and ground wire close together is really important to reduce the noise pickup in this cable. If the signal out and ground wires were farther apart, the ambient rf noise in the room picked up in the wires would be a significant problem.

The small header pins that plugs into a solderless breadboard make a secure connection to the solderless breadboard but is easy to insert and remove when changing out sensors.

The three wires connect to each sensor are color coded in the conventional way:

- *5 V: red*
- *Gnd: black*
- *Sensor out: white*

Figure 5.4 shows an example of the solderless breadboard with the three-pin headers plugged in for each of the four sensors.

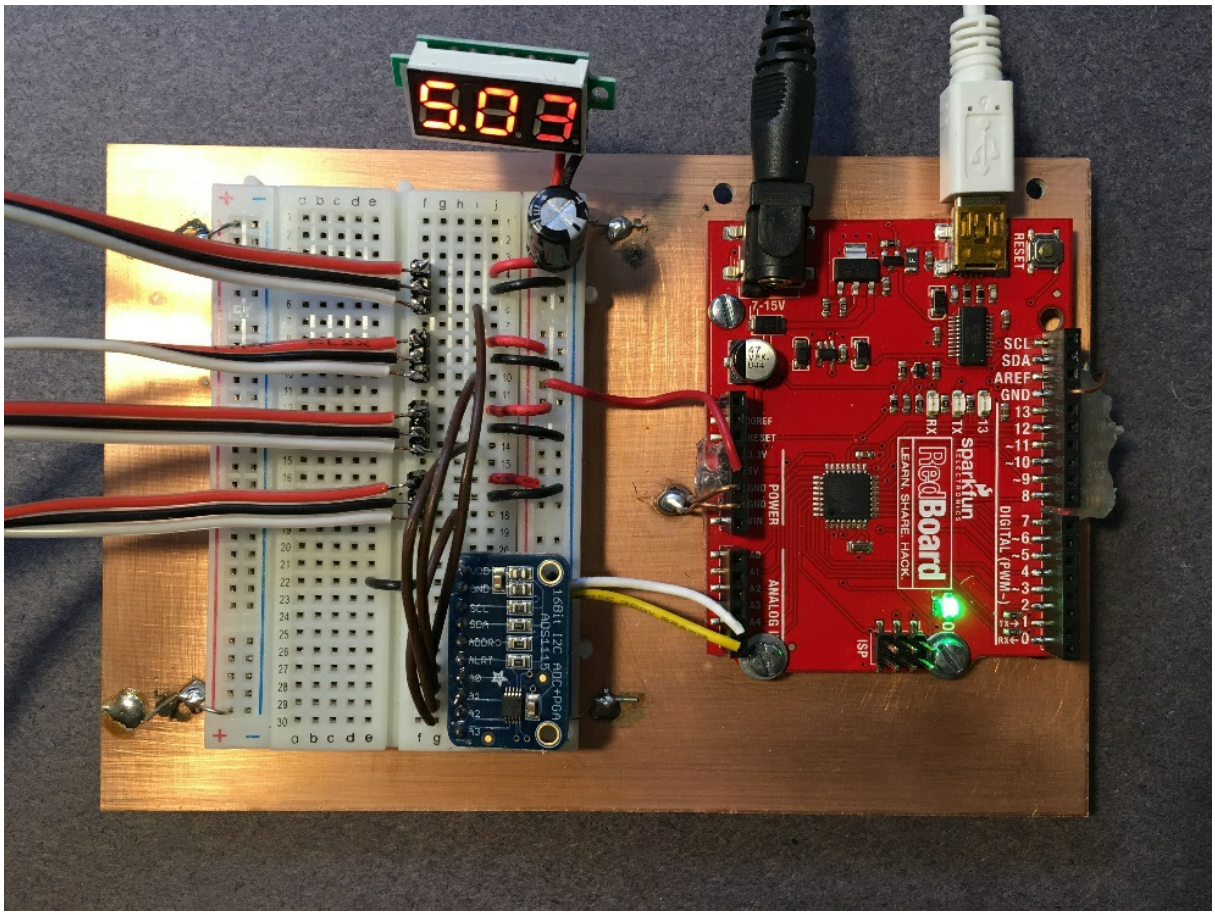


Figure 5.4 Example of a solderless breadboard with four 3-pin headers connected to the TMP36 sensors.

This is an example where an Arduino with an adjacent half-size solderless breadboard makes a convenient configuration. As we demonstrated in the examples measuring precision voltages, using just jumper wires and no ground plane in the solderless breadboard, we can still achieve a noise level of < 10 uV rms. This configuration of a solderless breadboard mounted adjacent to an Arduino board is available, for example, in the Sparkfun Inventors Kit.

The ADS1115 intrinsic noise level in a simple solderless breadboard configuration corresponds to a temperature noise level of 0.1 degC/mV x 0.01 mV = 1 milli degC.

The rated power supply rejection ratio of the TMP36 is about 0.05 degC/V of power rail. If the power rail noise is 0.01 V, about what the USB hub voltage noise has been measured, the impact on the temperature measurement would

be 0.05 degC/V x 0.01 V = 0.5 milli degC. This below any measurement noise floor.

This suggests the 5 V supply voltage is not critical. The TMP 36 would work just as well with the 5 V rail from the USB hub or using the on-board 5 V LDO as when an external 7-12 V supply is used.

However, as a good habit, when doing any data acquisition, if it is convenient, the external power supply should be used so the on-board LDO will supply the 5 V rail.

An external 9 V power supply was used for all of these measurements, which means the on-board 5 V LDO was used.

In this specific implementation, I used a solderless breadboard mounted on a ground plane, as described in HackingPhysics Journal, vol 1, no. 3, Oct 2020. It was used in this example because it was available, not because the ground plane was needed.

The header pins are inserted into the solderless breadboard and are firmly held in place. The ribbon cable is about 2 feet long, long enough to reach from wherever is being measured to the solderless breadboard connections. Using the header pins, it makes adding or replacing a temperature sensor probe very easy.

The ADS1115 on the solderless breadboard measures all four of the TMP36 sensors, from channel A0 to A3.

At the sensor end, the three wires are soldered to the sensor itself. To enable the sensor to be immersed in water or other liquids, the wires are potted in 5-min epoxy or in hot glue.

For mechanical support, the potted sensor and wires are inserted in the end of a straw and then potted with hot glue. Figure 5.5 shows the TMP36 mounted into the potted straw.

Figure 5.5 Close up of the TMP36 sensor with three wires soldered on the end and potted in hot glue then mounted inside a straw.

5.3 First temperature measurements

The sketch developed for the voltage measurements was used to collect all the temperature measurements.

Each channel is measured as a single-ended measurement so that up to four channels can be measured. Each channel is measured at 108 SPS, interleaved, and averaged for 60 power line cycles, or 1 second.

The 1-second averaged voltages are converted into temperatures and then plotted directly into excel using PLX-DAQ v2. This technique is described in detail in Science Experiments with Arduinos.

The voltages are converted into temperature using the default calibration terms, 0.1 degC/mV and -50 degC, and then into degF. The measurements are taken for 1 hour and plotted in excel.

As an example of the measured temperature of the air in my lab over a 1-hour period from the four different sensors is shown in Figure 5.6.

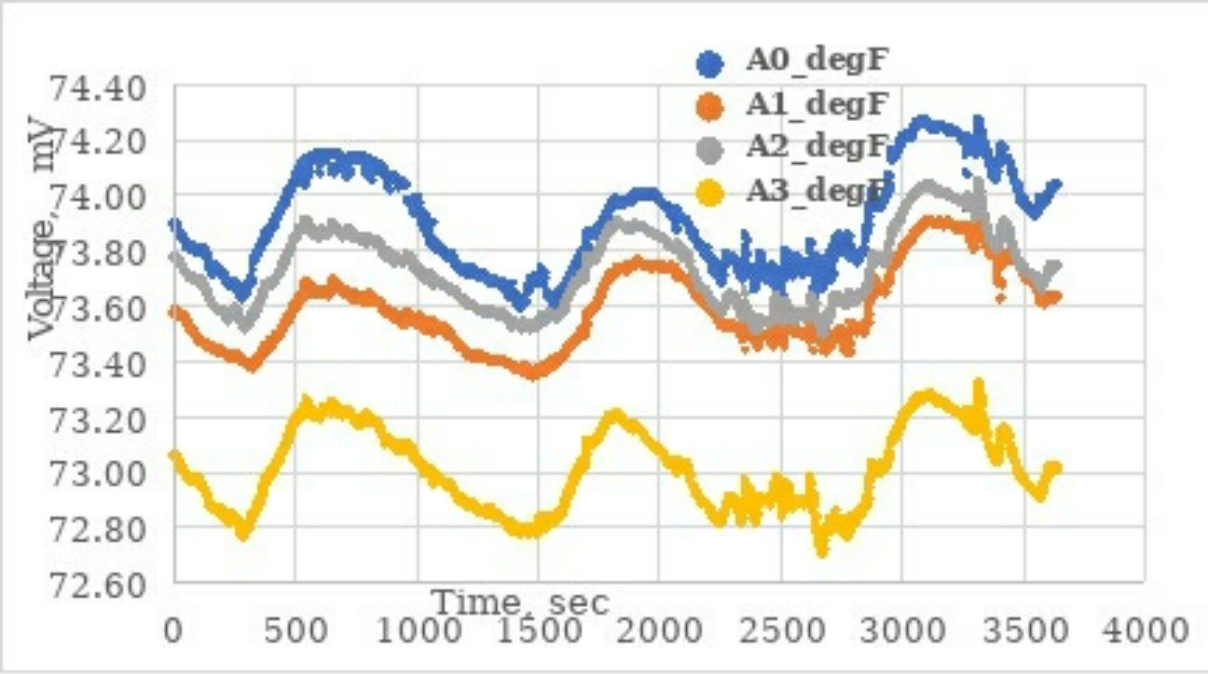


Figure 5.6 Measured air temperature in my lab from the four sensors using the simple solderless breadboard and one ADS1115 module. The scale is 0.2 degF/div or 1.8 degF full scale.

Even though each of the sensors are in the same environment, they read a slightly different temperature. This is due to the rated absolute accuracy of ±1 degC, which is ±1.8 degF. This is a range of 3.6 degF peak to peak.

It is remarkable that using the same, default calibration values for each sensor, the agreement between the four sensors are all within a range of ±0.5 degF. This is well below the specified range.

Notice also that this shows a very small temperature variation in the ambient room in my lab. The temperature varies within a range of about 0.4 degF. This is all due to the thermostat control in the house.

This is an indication of the potential in the ability to track small temperature changes using this simple, low cost temperature measurement system.

5.4 Using a USB cable for remote sensing

Ideally, to reduce the noise from the environment, we want to keep the length of the ribbon cable as short as possible. This will reduce the possible noise pick up that can get between the TMP36 analog voltage source and the

ADS1115 module where the voltage is turned into a digital signal.

And, for lowest noise, the analog signal should be measured as a differential signal between the output voltage and its local ground.

As a compromise, we are using a single-ended measurement. This will introduce about 10 uV rms noise in the measurements due to the ground bounce noise. This is well below any important noise threshold.

Using a ribbon cable keeps the signal and its return-ground line in close proximity so there is only a small area for inductively coupled noise.

In the noisiest environments or when the distance between the sensors and the Arduino is longer than a few feet, a lower noise alternative is to keep the ADS1115 close to the sensor. Once the information is converted into a digital signal, the I2C bus between the Arduino and the ADS1115 can be as long as necessary.

The connection between the ADS1115 to the remote Arduino is just the +5 V, gnd and the SCL and SDA lines. These are DC and digital signals, just four lines.

To make the connections between the Arduino Uno board and the remote ADS1115, we need a cable that has at least four connections. The simplest solution is to use a USB 2.0 cable. It has four wires and a ground shield.

The signals on this cable are either DC or digital signals. These are very insensitive to any sort of analog noise which might be picked up from the environment. This means the cable with only DC and digital signals can be made very long if needed.

The simplest way to use a USB cable is to take a USB extension cable with a female connector on one end and a made connector on the other and cut it in half. We end up with bare wires on one end and a male or female USB connector on the other end.

The bare wire end gets connected to the Arduino and its I2C bus connections and the other half of cable to the ADS1115 and its I2C bus connections. We just use the matching USB connectors to connect or disconnect the cables.

The wire connections are simple:

- *Red is 5 V power*

- *Black is gnd*
- *Green in the SCL*
- *White is the SDA*
- *The cable shield is connected to gnd*

The best part of this is we don't have to build a custom cable/connector system to connect to the Arduino and the ADS1115. This way, we can just connect the USB connectors to each other to make the connection or disconnect them to provide some mobility.

If we want to go a longer distance, we just insert a standard USB extension cable between the male and female ends of the USB cable.

This makes for a simple, robust, easy to construct modular system. An example of the connections at the two ends between modules is shown in Figure 5.7.

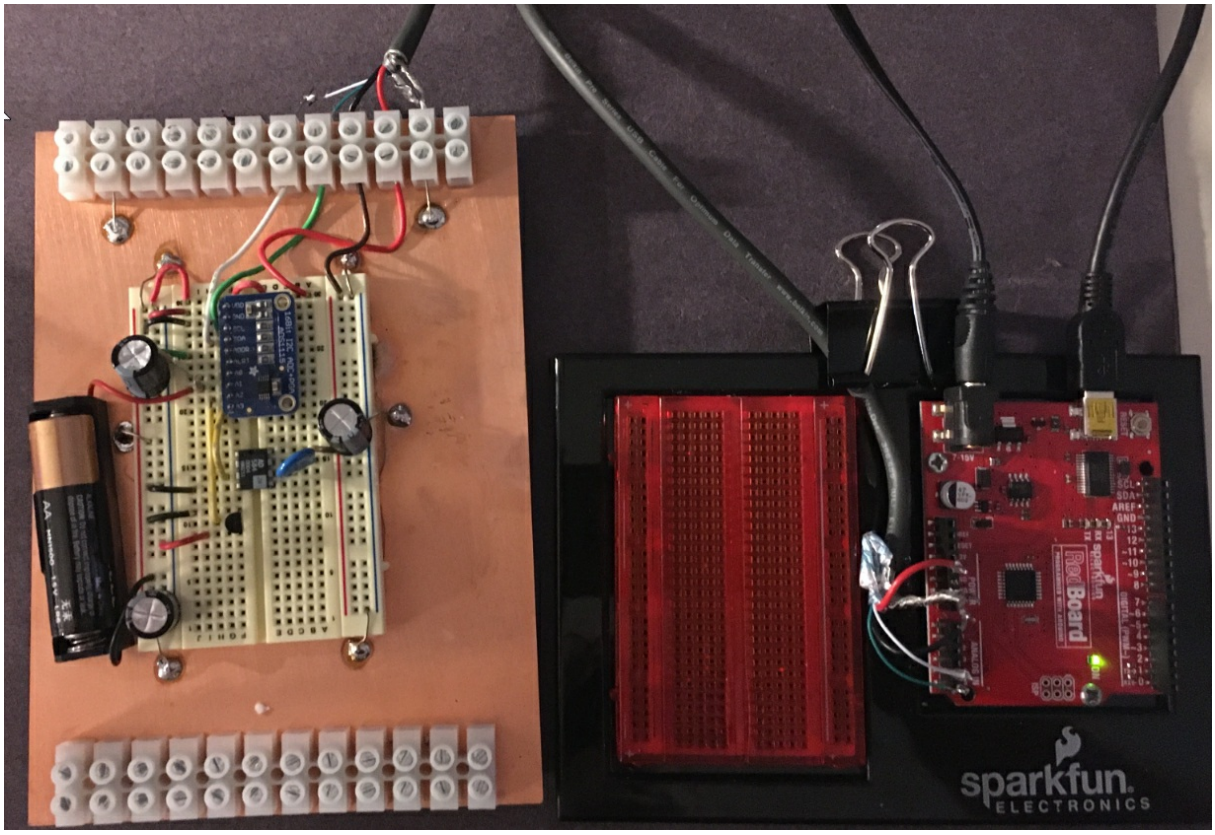


Figure 5.7 Example of two modules connected by a USB cable to connect the power, gnd and SDA and SCL lines.

Chapter 6. Voltage Noise in the TMP36

In the course of high-performance measurements of the TMP36, I noticed a significant high frequency noise source that causes glitches in the temperature measurements. When looking at uV level signals from the TMP36, it is inevitable some of this high frequency noise will affect the low frequency voltage measurements which we interpret as a temperature.

The first step in precision temperature measurements is to reduce this noise.

6.1 Measuring the rf noise in my lab

Eventually, I traced the source of noise to ambient rf pickup in my lab. I tried to turn every device off in the lab to see what affected the noise but could not find the specific source. It was coming from somewhere else in the house.

There were two specific transient noise sources. One was periodic with 120 Hz and tied to the line voltage phase. The other was asynchronous with the line phase and at 90 kHz.

Each of these noise sources had a signature of transient ringing with very fast frequency components of the ringing, on the order of 25 MHz.

The temperature sensor was mounted to a 2-foot length of three conductor ribbon cable with a 3-pin header to plug into the solderless breadboard. This cable acted like an antenna and picked up the rf noise in the lab.

To emulate the pick-up from this cable, I built an identical cable, but with three header sockets on the sensor end so I could change the impedance of the end. This was a dummy cable just to act as a similar antenna. This configuration is shown in Figure 6.1.

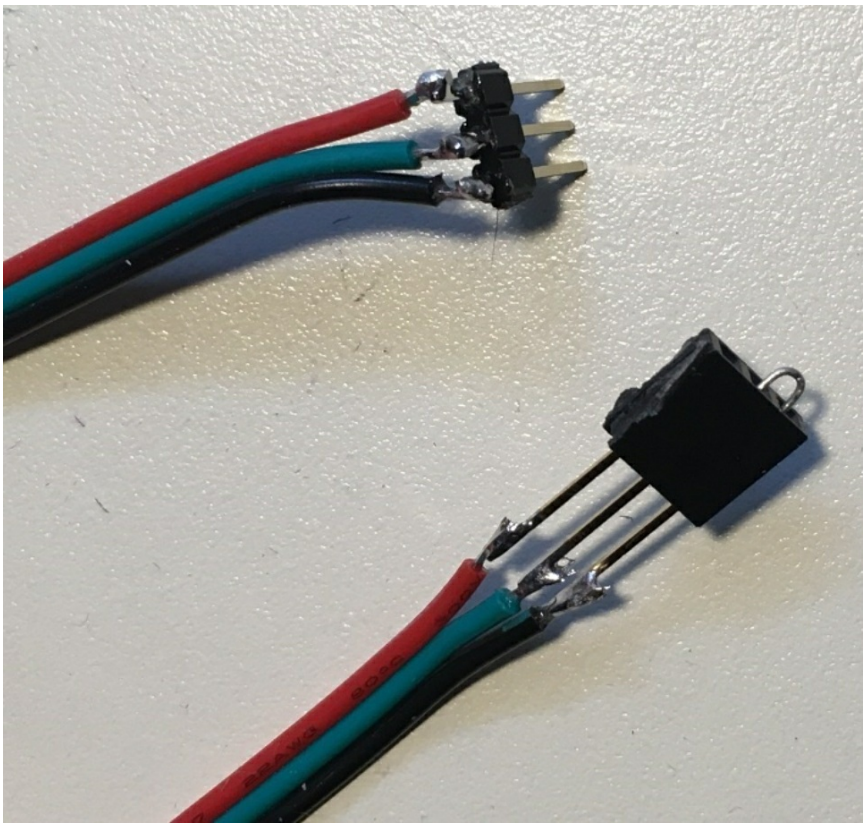


Figure 6.1 The ends of the dummy cable showing the 3-pin header that plugs into the solderless breadboard and the short at the far end where the sensor would go.

For this length of cable, the frequency of one wavelength fitted along the cable length was 500 MHz, for ½ wave, it was 250 MHz and for ¼ a wavelength, it was 125 MHz.

All of these frequencies are above what I can measure with my 50 MHz bandwidth AD2 scope. This means that all the measurements will be from an antenna that is electrically short. It will be sensitive to electric fields when the cable is high impedance and sensitive to magnetic fields when the cable is shorted at the sensor end.

With the header socket on the sensor end, I am able to explore the source of the pickup: electric field or magnetic field by keeping the far end open or shorted.

Also, to create an environment as close to the final measurement, I used the same solderless bread board into which the sensors would plug.

To get a measure of the noise of the AD2, I shorted the two V+ and V- inputs

to channel 1 to gnd on the solderless breadboard of the AD2 to measure the magnetic pick up noise from the short twin lead cable from the AD2 to the DUT. This is the intrinsic noise in the AD2.

There is no measurable pick up above the noise floor, as shown in Figure 6.2.

Figure 6.2 Noise floor of the AD2 with the inputs shorted together and otherwise floating. The voltage scale is 2 mV/div.

There is no change in the pickup noise when the two leads of the AD2 are shorted together on the solderless breadboard of the sensor board.

When the leads have a high impedance, there is a very different behavior. In this configuration they are sensitive to electric fields. Figure 6.3 shows the measured voltage noise on the open leads when they are plugged into floating pins of the solderless breadboard.

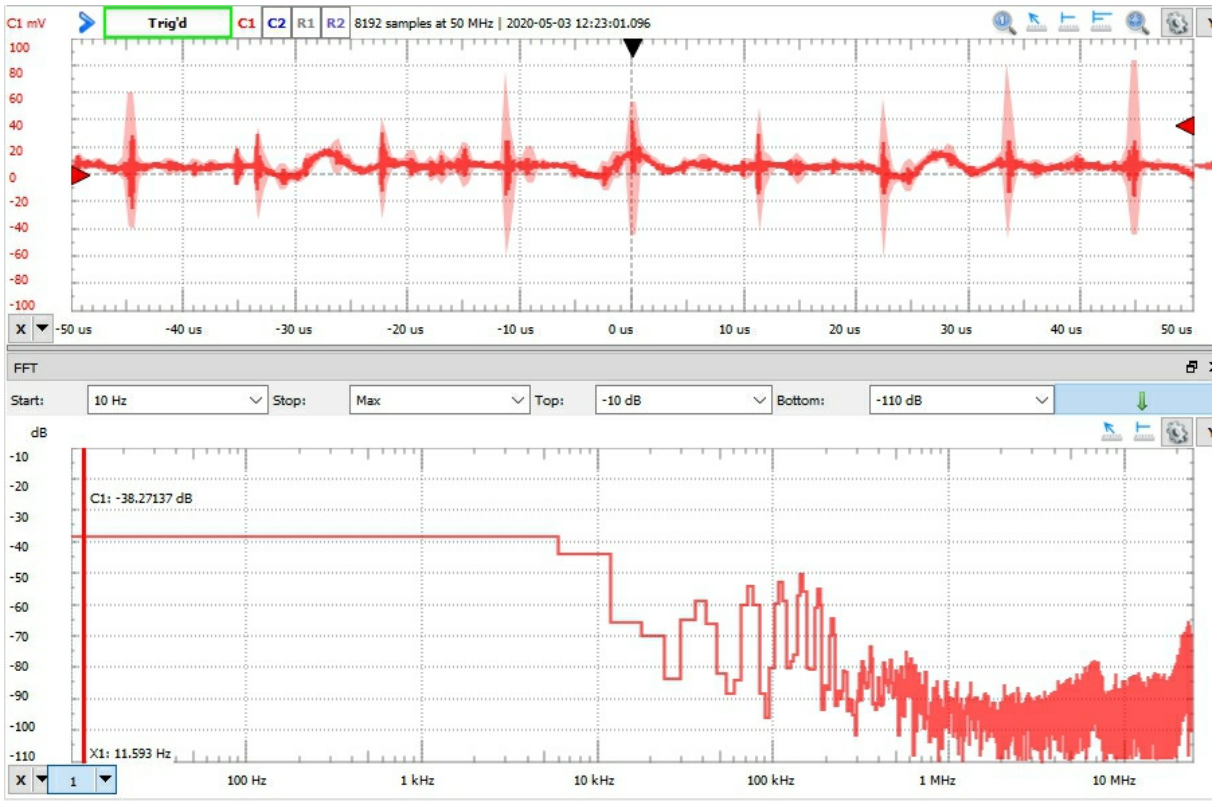


Figure 6.3 Pickup noise from open leads.

There are a few different signatures in this pickup noise. There are high frequency bursts, spaced about 11 usec apart. This is a frequency of 90 KHz. These frequency components are apparent in the spectra.

It is important to note that not always will a periodic frequency component appear in the spectrum at the repeat frequency. If the pulses are bipolar, so that they average to zero in a fraction of a cycle, then their discrete Fourier components may average to zero. This is why it is important to also view the time domain behavior.

In addition to the frequency component at 90 kHz, there is also a component that is synchronous with the 90 kHz component with a period of about 30 usec or a 30 kHz.

The glitch waveforms are very fast ringing pulses with a frequency of about 25 MHz. This is evident when I used a shorter time base and a higher frequency limit in the spectrum. Figure 6.4 shows this spectrum of 25 MHz ringing.

Figure 6.4 Measured noise pick up with the leads open, on a shorter time scale. Note the small peaks at about 25 MHz. This is the frequency of the ringing transients.

When the dummy cable was added to the input to the scope on the solderless breadboard, with the far end shorted, so we are sensitive to magnetic fields, the signature measured was very similar. There was a strong periodicity at about 90 kHz with high frequency noise on the order of 25 MHz. This is shown in Figure 6.5.

Figure 6.5 Extension cable added to the scope input, with the far end shorted. This is similar to the impedance of the TMP36 when it is powered on.

Just as a second test, I measured the noise in the AD2 scope with the dummy extension cable added, shorted at the far end, but with the Arduino board powered on. The noise signature and levels were nearly identical. There is no impact on the rf pick up when the Arduino is powered on.

6.2 Measured noise on the TMP36 sensor

The dummy cable was disconnected and the TMP36 cable was plugged in. The noise signature with the TMP36 powered on and shorting the far end of the cable was a little bit different from just the short in the dummy cable. There is still a 90 kHz periodicity to the noise pickup, but it has a different signature. This is shown in Figure 6.6.

Figure 6.6 Measured noise with the TMP36 cable plugged in, in red, and the previous measurement of the dummy shorted cable, in grey. The voltage plots are on the same scale.

The noise with the 90 kHz periodicity has been rectified by the nonlinear output impedance of the TMP36 creating a larger frequency component at roughly 90 kHz and higher harmonics.

The nonlinear nature of the TMP36 output has changed the signature of the 90 kHz ambient rf pick up noise.

It is interesting to note that this noise level is about 100 mV peak to peak. This is huge. But there is more.

Since the noise we are seeing is all pickup noise, if the coupling to the ribbon cable of the TMP36 increases, by my touching it for example, there will be enhanced pickup. Touching the outside of the cable induces more electric field coupling and shows up as large voltage transients that are synchronous with the line frequency, but at 120 Hz. Figure 6.7 is an example of this transient voltage noise on the TMP36.

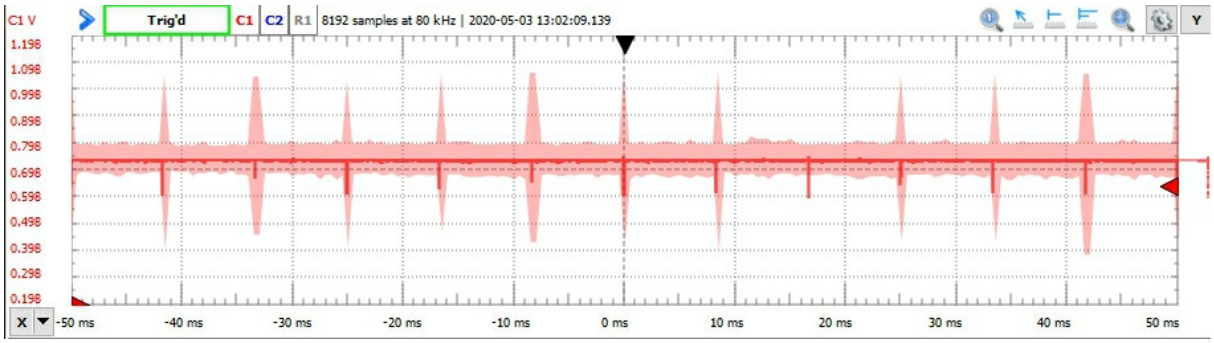


Figure 6.7 When I touched the outside of the ribbon cable, the electric field pick up noise increased and shows the 120 Hz component, synchronous with the line frequency. Note the scale is 100 mV/div.

In this example, when I touched the outside of the cable to the TMP36 sensor, the transient noise was as large as 600 mV peak to peak. This is huge.

This means that the pickup noise can vary as the environment varies, as I move closer or farther to the sensors, or as other sources of rf noise turn on. In worst case, this rf pick up noise that appears at the end of the TMP36 cable that is read by the ADS1115 channel can be more than 600 mV peak to peak.

While rf noise generally is bursts of rf sine waves with an average value of 0, the nonlinear nature of the TMP36 output impedance and possible saturation of the ADS1115 scale can rectify this rf pick up a little bit. Since we are looking at slowly changing voltage on the order of 10 uV, even a 0.1% rectification of a 600 mV signal will result in a 600 uV offset.

6.3 Reducing the rf pick up noise in the sensor

There are three simple fixes to reduce this problem. The first is to keep the cable lengths in which the analog voltage is traveling, as short as practical. The pickup magnitude will scale with the cable length.

Second is to use a shielded coax cable. Some of this noise can be eliminated with the electrostatic shield of a coax cable that is connected to ground at the end at the solderless breadboard.

A third alternative is to filter this rf noise out at the ADS1115 receiver using a capacitor. The datasheet says the TMP36 can drive a 10 nF capacitor. Larger values are possible, they just were not tested. In order to get the maximum filter, I added a 100 nF capacitor to the output of the TMP36, at

the end of the ribbon cable on the solderless breadboard.

This produced a very dramatic reduction in the rf pickup as measured by the AD2 scope. This noise level with the 100 nF capacitor added, is shown in Figure 6.8.

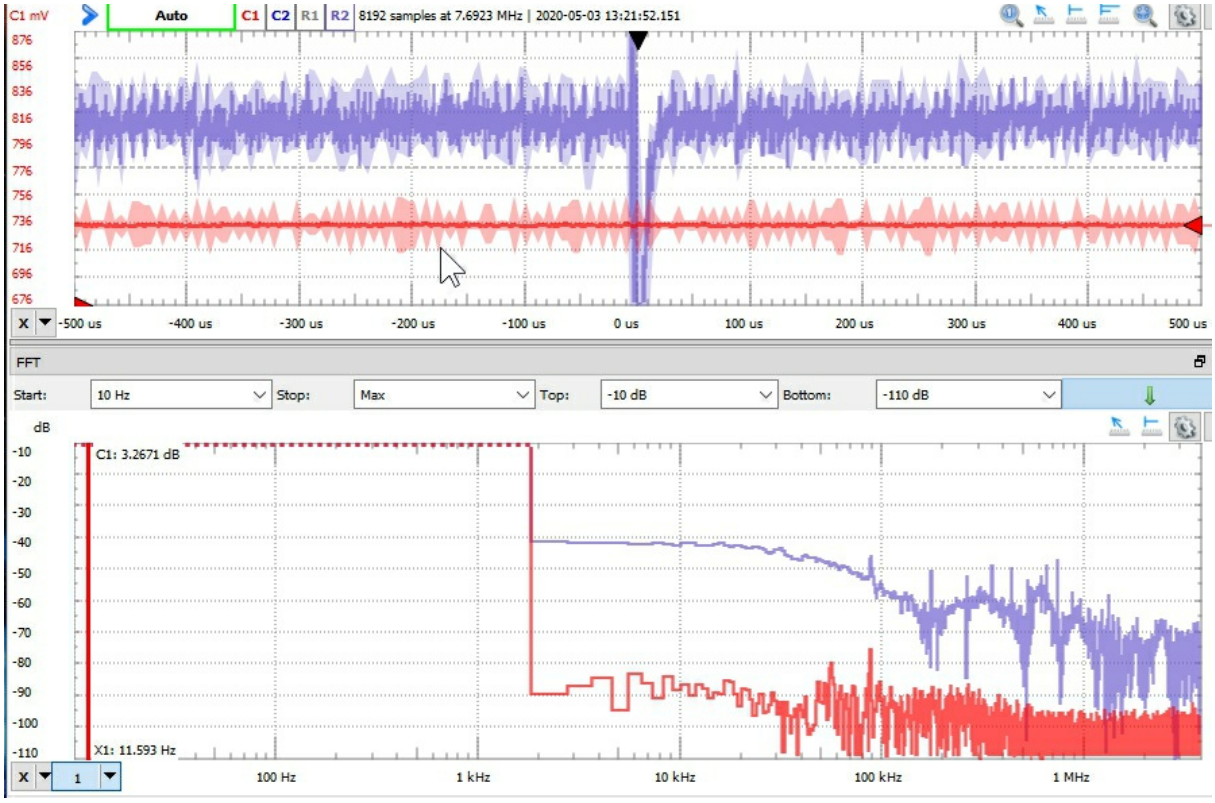


Figure 6.8 Noise in the TMP36 with a 100 nF capacitor across the input, in red, compared to the noise with no capacitor, in purple.

The capacitor filtered most of the rf pickup, but not all of it. Since the amount of rf pickup can vary depending on the local environment, it may change on a daily basis.

The TMP36 has been reported as being noisy. I think the real cause of this noise is the nonlinear output impedance of the sensor and the rf pickup of any wire connections between the sensor and the ADC used to measure it. This causes the analog voltage measured by the ADC to be slightly sensitive to the rf pickup.

For the most sensitive measurement, be aware that unless a shielded cable is used, there will be some rf pickup which may be translated into a DC component. As the rf pickup component changes, due to either changes in the

source or in the coupling to the ribbon cable, this may translate into very small temperature glitches.

Figure 6.9 shows the configuration of the solderless breadboard with 100 nF capacitors at the output of each TMP36 cable in the solderless breadboard.

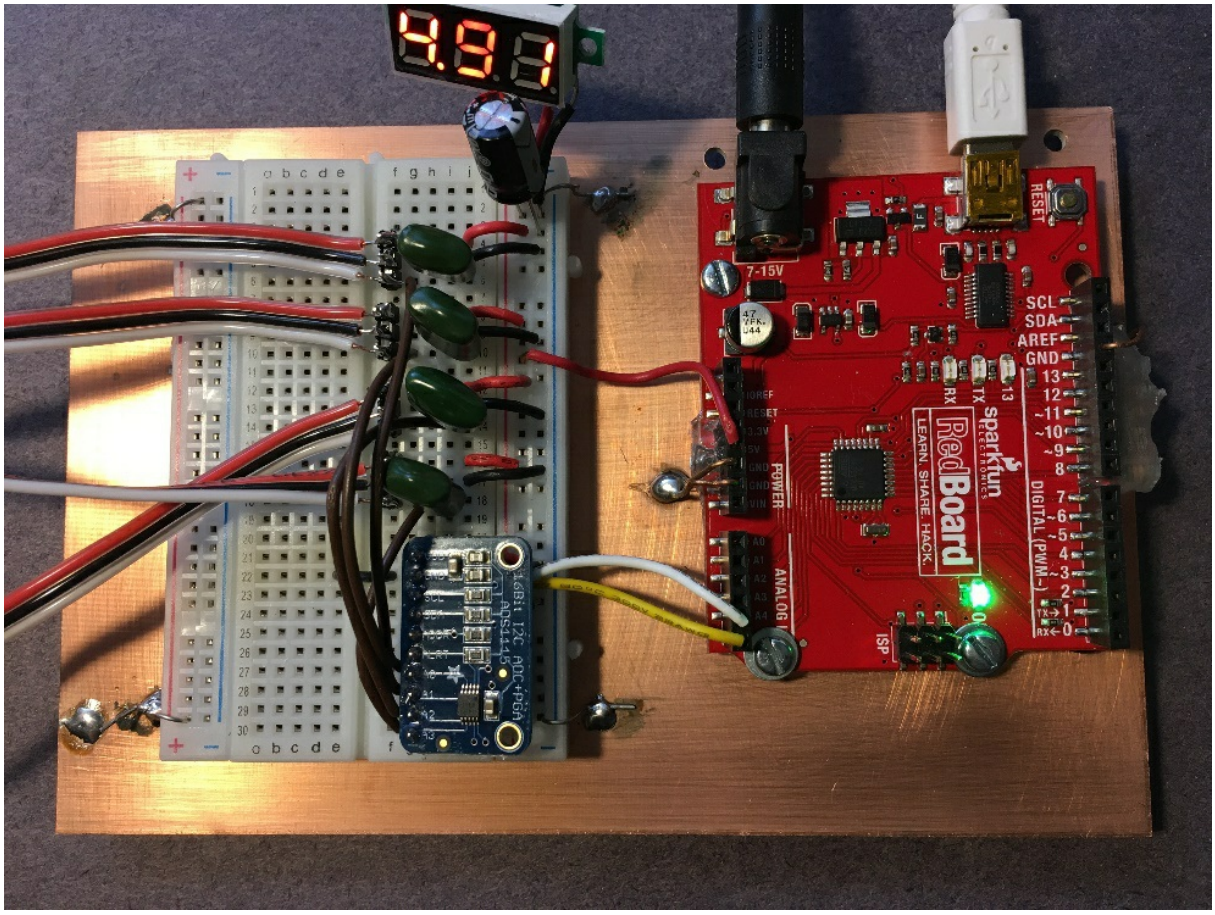


Figure 6.9 Recommended configuration using 100 nF capacitors (dark green) at the output of the TMP36 to suppress the rf pick up noise.

6.4 The impact of reducing rf pick up

As an experiment to evaluate the impact of adding the 100 nF filter capacitor, we removed the capacitors on channels A2 and A3 of the four-channel temperature measurement system. The channels A0 and A1 had 100 nF capacitors.

I connected the sensors and left them in the ambient air. The temperature of all four channels was measured using the standard sketch so that every

second, the averaged temperature of each channel was printed into excel and plotted.

Periodically, while I measured the ambient temperature, I grabbed hold of all the cables and held them. This would increase the rf coupling noise from me to the cables and would cause a small temperature glitch.

I expected channels A0 and A1 to be less sensitive to these glitches since they had the 100 nF filter capacitors, while channels A2 and A3, without a 100 nF filter capacitor, would show some sensitivity.

This is exactly what I observed. Figure 6.10 shows the measured temperature during a 5-minute period where I grabbed the cables a few times. In this plot I am comparing just A0 and A3, with and without the filter capacitor. The difference in glitches is clear.

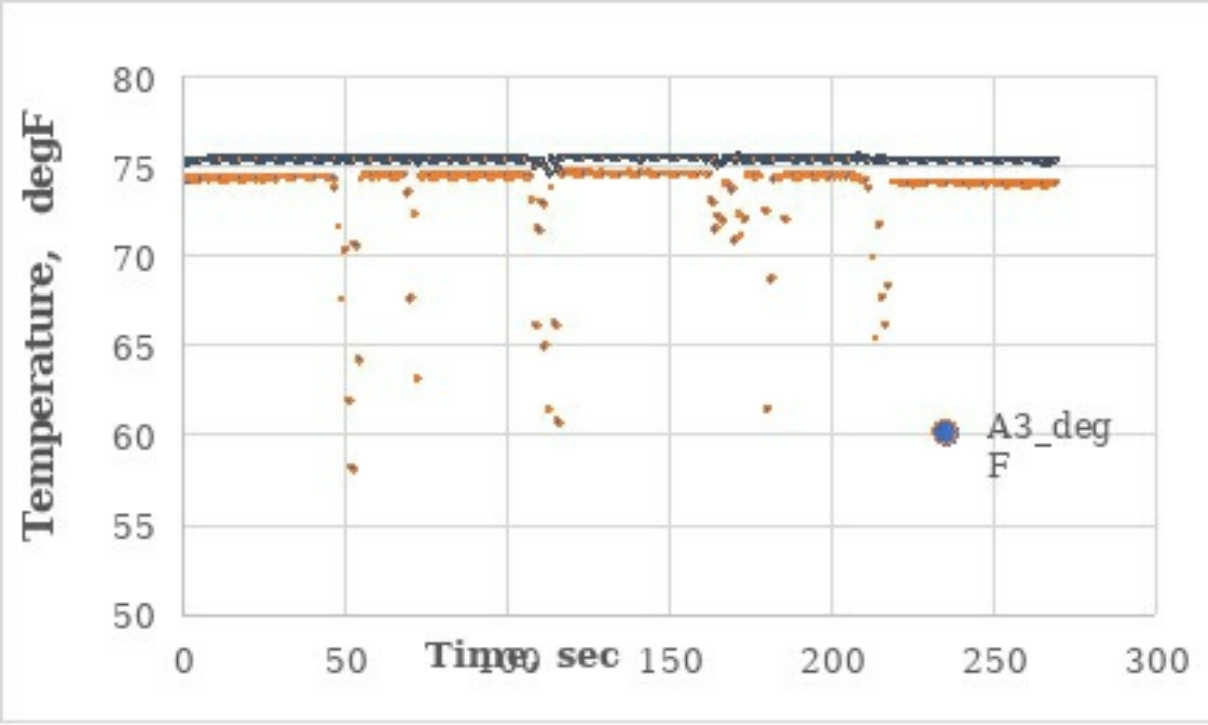


Figure 6.10 Comparing the temperature measurements of channel A0 with a 100 nF filter capacitor and channel A3 that did not have a filter capacitor, while I periodically grabbed the cables to induce more rf pickup.

There is still some sensitivity to the DC temperature measurement from rf pick up even with the 100 nF capacitors, but it is dramatically reduced.

By comparison, channel A3 with no capacitor, shows glitches on the order of

15 degF when I touched the cable. Channel A0 that has a 100 nF filter capacitor shows glitches that are about 0.8 degC. This is a reduction by a factor of 20 in the sensitivity to rf pick up.

It should still be noted that with this filtering, and the averaging, there is still some sensitivity to the temperature readings from the environmental rf pick up. For the most sensitive measurements, a coax cable to the sensors should be used and hands should be kept off the sensor cables.

In most of the following measurements, the 100 nF filter capacitor was used for each channel on the solderless breadboard.

Chapter 7. Calibrating the temperature

The rated absolute accuracy of the TMP36 is ±1.8 degF. In practice, I have found that using the default values of the calibration terms of the temperature sensors, the agreement between four different random sensors is actually within ±0.4 degF.

To get a measure of the absolute accuracy, I used a simple digital thermometer as the reference. The Kizen meat thermometer is rated at an absolute accuracy of ±1 degF and responds in just a few seconds. It has a digital LCD that displays temperature to within 0.1 degF.

This can be used as a secondary reference so any stable temperature can be used to calibrate the TMP36 sensor.

7.1 Accuracy vs precision

There is a difference between the absolute accuracy of the temperature and the relative accuracy, or precision, of temperature measurements.

The absolute accuracy is basically a measure of how well a measurement of temperature agrees with a NIST traceable measurement of temperature.

To measure the absolute accuracy requires a temperature standard, or a secondary temperature instrument that has a NIST traceable accuracy.

The relative accuracy or precision of a temperature measurement is a really about the reproducibility of a temperature measurement and how well different, uncalibrated sensors would agree to the same temperature environment. This is a measure of how small a change can be seen and how reproducible are temperature measurements of the same temperature.

The relative accuracy is related to how close in temperature multiple sensors will read when in the same thermal environment. They may not all be accurate, but they can be very close in their values.

7.2 Calibrating the TMP36 with two temperatures

The absolutely accuracy of the TMP36 is rated at ±1 degC or ±1.8 degF.

While the relative agreement between multiple sensors and the Kizan digital thermometer is within 0.3 degF for these four sensors, this may be a special case for these specific sensors in this batch from the vendor.

The accuracy of the voltage measurement is as accurate as the ADS1115 measurement. This is better than 1 mV absolute accuracy. This corresponds to about 0.1 degC. This means that the inherent absolute accuracy of a temperature measurement could be to within 0.1 degC if the conversion between the voltage and temperature was accurate.

If we assume the temperature conversion from a voltage for each sensor is linear, the conversion between a voltage and a temperature requires two figures of merit:

- *The slope of degC_per_mV*
- *The T0 temperature when mV = 0*

The conversion is:

Temp_degC = mV x degC_per_mV + T0_degC

The T0_degC value will be a negative value.

These two parameters can be measured for each sensor from the voltage on the sensor at two different temperatures. The Kizan digital thermometer was used to measure the temperature while the ADS1115 was used to measure the voltage on the sensor.

Two different water baths were used as the temperature sources.

From the two different voltages, V1 and V2 and the two different temperatures, T1 and T2, the two figures of merit were calculated for each sensor as

degC_per_mV = (T2 -T1)/(V2 – V1)

T0_degC = T2 – V2 x degC_per_mV

Or equivalently,

T0_degC = T1 – V1 x degC_per_mV

The conversion coefficients offered in the data sheet for the TMP36 are 0.1 degC/mV and -50 degC.

For the two different temperatures, I used an ice bath with a temperature of 31.6 degF and ambient air, at 74.8 degF, as measured by the Kizan digital thermometer. I recorded the voltages on each sensor at these two temperatures, after the voltage had stabilized.

Based on the voltage on each sensor at the two temperatures, I measured the following calibration terms:

sensor	**T0_degC**	**Slope degC_per_mV**
Data sheet	*-50 degC*	*0.1 degC/mV*
Sensor 0	-49.5	0.101
Sensor 1	-50.1	0.099
Sensor 2	-49.7	0.098
Sensor 3	-49.3	0.1006

It is remarkable that the slopes were within 1% of the data sheet values and the offset temperatures were within about 1% of the data sheet values.

If it is necessary to have all the temperature readings be the same or nearly the same, it is possible to use the data sheet value for the slope for each sensor, and just use the offset temperature as a unique calibration term for each sensor.

When the sensors are in the same thermal environment, at T1, and we measure a voltage, V1, then the offset temperature value for each sensor is

T0_degC = T1 – V1 x degC_per_mV

This will force all the sensor temperature values to match at the starting temperature of T1.

As a test of this, I immersed all four sensors in a cold-water bath inside an insulating cup and allowed the water to slowly heat up. I used the custom calibrated slopes and temperature offsets for each sensor.

The temperature of these four sensors in the same bath of cold water is shown

in Figure 7.1. The agreement in temperature for each sensor is seen to be within 0.05 degF. This is a remarkable agreement. It is not expected to last over a wide temperature range, as the sensors may not have a perfectly linear temperature response.

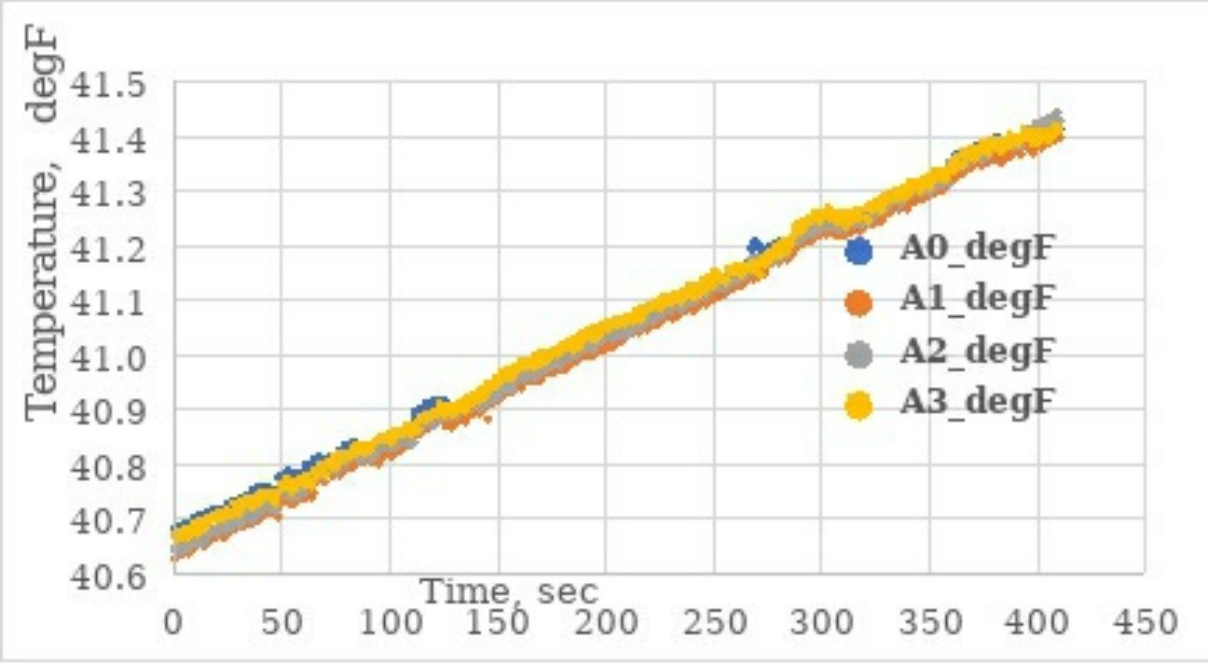


Figure 7.1 Measured temperature on four sensors immersed in a thermos of initially ice water, allowed to heat up. The temperatures were matched initially.

It is interesting to note that at various times during the 1-hour monitoring period, there were correlated changes in the temperature in each of the sensors on the order of less than 0.05 degF.

Chapter 8. Stable temperature environments

Now that we have a way of monitoring the temperature of four sensors with a resolution of less than 0.01 degF, we can evaluate how constant some temperature environments actually are.

8.1 Temperature stability of an ice bath

When comparing the relative temperature measurements of multiple sensors in the same environment, the most important criteria is to make sure each sensor is really at the same temperature.

One way is to use a circulating water bath. In the first example, I filled an insulting cup with water and ice and used a small magnetic stirrer so that the water circulated. The four TMP36 temperature sensors were immersed in the bath.

Into this bath, I also inserted the Kizan digital thermometer to get a reading of the temperature of the bath. This measurement system is shown in Figure 8.1.

Figure 8.1 An ice bath with circulating water, ice, four sensors and the digital thermometer.

The temperature read by the thermometer was -0.1 degC. This is 31.8 degF.

I also used the custom calibration coefficients for each sensor and immersed them in this same environment. They were measured over a 1-hour period. Figure 8.2 shows these measurements from four different, calibrated sensors.

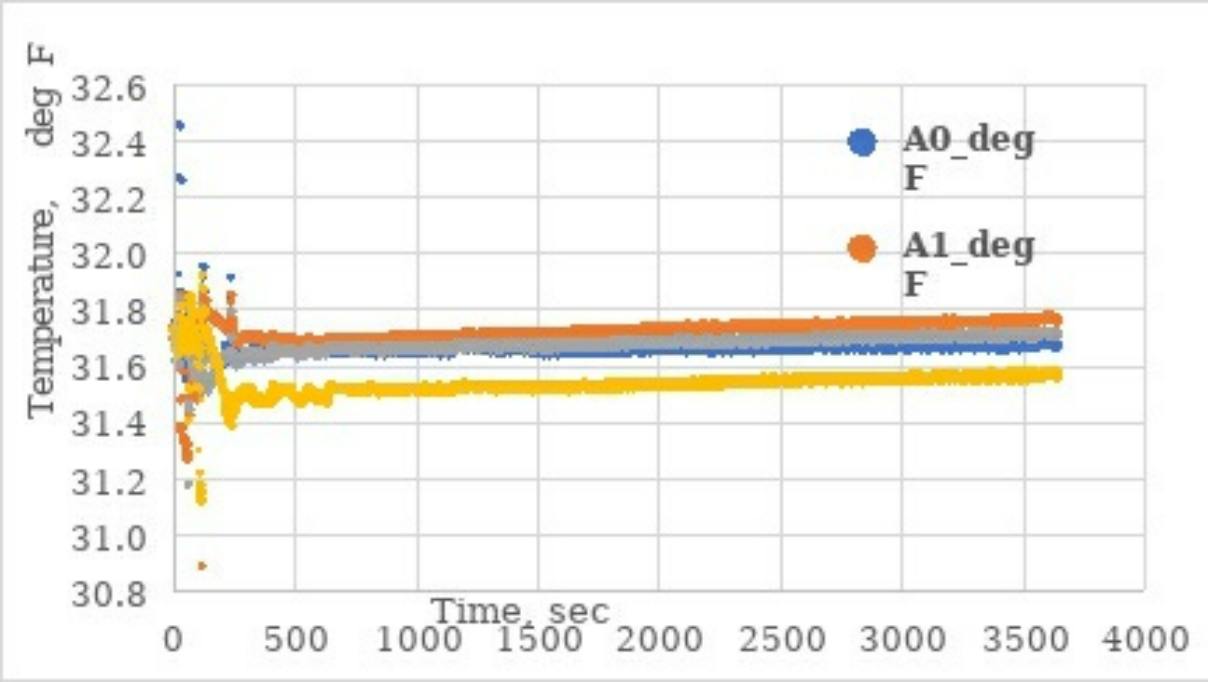


Figure 8.2 Measured temperature with all the sensors immersed in the ice bath each with a custom calibration.

The temperature of the bath was measured as 31.8 degF, with an absolute accuracy of ±1 degF. It is remarkable that the agreement between all four sensors was within 0.3 degF. They differ from the 31.8 deg digital thermometer reading by 0.3 degF.

In this example, the sensors do not measure exactly the same temperature, even though they were each calibrated. A hint as to why is seen at the very beginning of the experiment.

In the first few minutes of measurements, the sensors were moved around in the bath. This is one reason the temperatures fluctuated. The other reason is that since I touched them, I increased the rf pick up. Even though they each had 100 nF filter capacitors, there was still some rf pick u sensitivity. This suggests that there were thermal gradients in the ice bath on the order of 1 degF.

If there really is a temperature variation inside the ice bath of as much as 1 degF, it is difficult to get a measure of the relative or absolute temperature readings to better than this. How do you know the sensors are seeing the same thermal environment?

For long time frame measurements, the temperature sensors were all placed in the same location in the bath to try to stay in the same thermal environment. The water was circulating the whole time and there was ice in the bath at all times.

During this 1-hour period, the temperature drifted by about 0.1 degF. The agreement between each sensor is within ±0.1 degC. This may be due to the thermal gradients or an intrinsic measure of the best we can do with relative calibration.

8.2 Ice bags in an insulating box

How stable can we get a temperature reading? I tried the approach of an ice bath and the temperature was stable to within about 0.1 degF over the period of 1 hour. But this requires immersion in a liquid bath. This approach may offer the most stable environment.

The ice acted as a thermal sink, keeping the temperature stable. If the temperature heated up a little, more ice would melt and absorb the heat and the temperature would drop.

In the next experiment, I wanted to create a stable temperate inside an insulating box in air. To increase the thermal mass and further stabilize the air temperature, inside the box I added bags of ice with a small circulating fan.

I’ve used this box in other experiments where a relatively stable thermal environment was important. Figure 8.3 shows the inside of the box with two bags of ice, the small fan and a small circuit board.

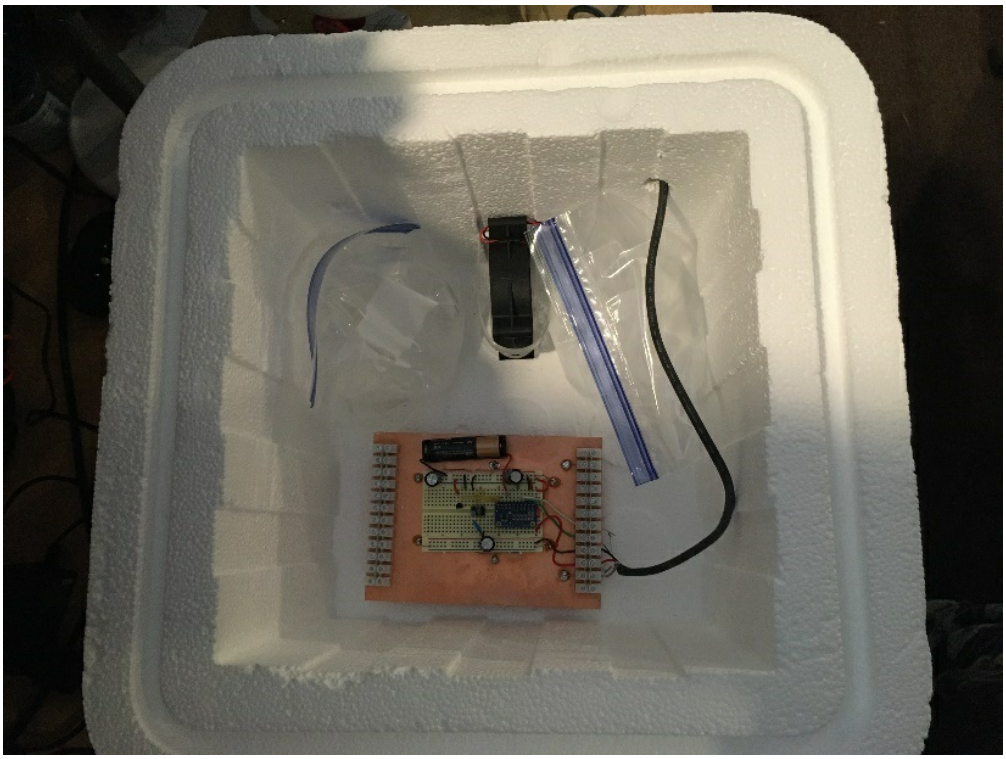


Figure 8.3 Inside the thick walled Styrofoam box with two bags of ice and a small circulating fan.

Into this box, I inserted the four temperature sensors and the Kizan digital thermometer. This set up with the connection between the four sensors and the solderless breadboard that contains the ADS1115 module is shown in Figure 8.4.

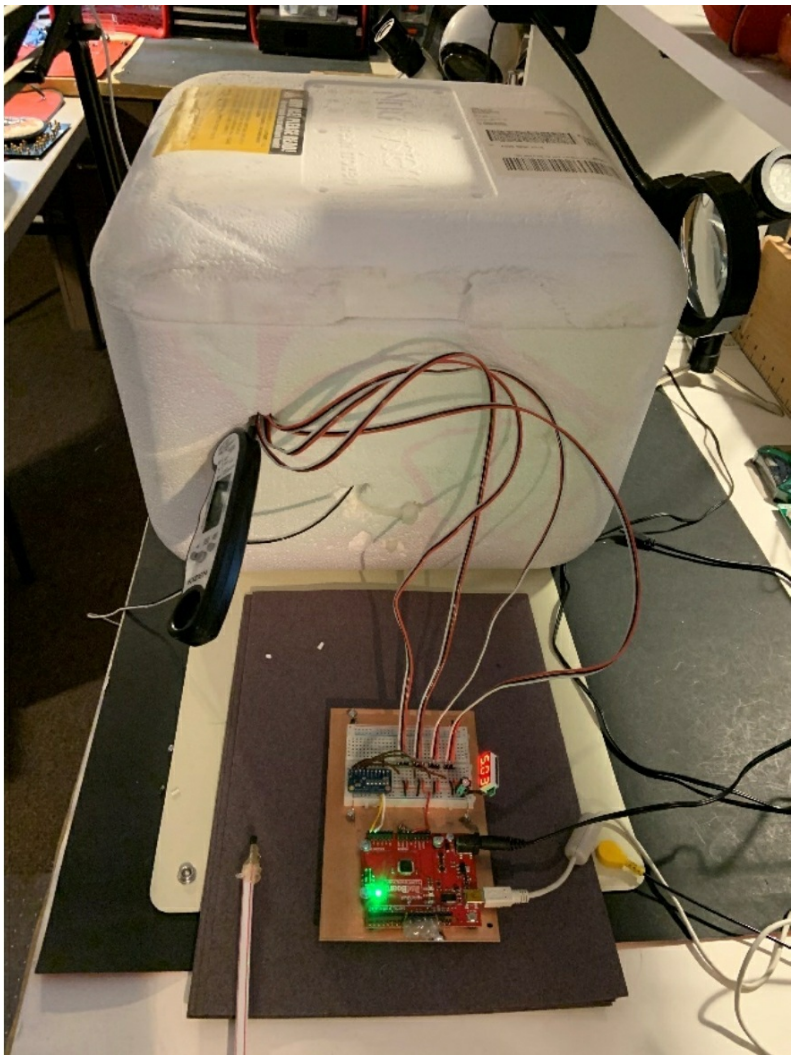


Figure 8.4 The insulating box with the four temperature sensors monitoring the ice bag-cooled air.

Once I inserted the bags of ice in the box and let them cool down the air, I could measure the equilibration time. Figure 8.5 shows the measured temperature in the box after adding the ice.

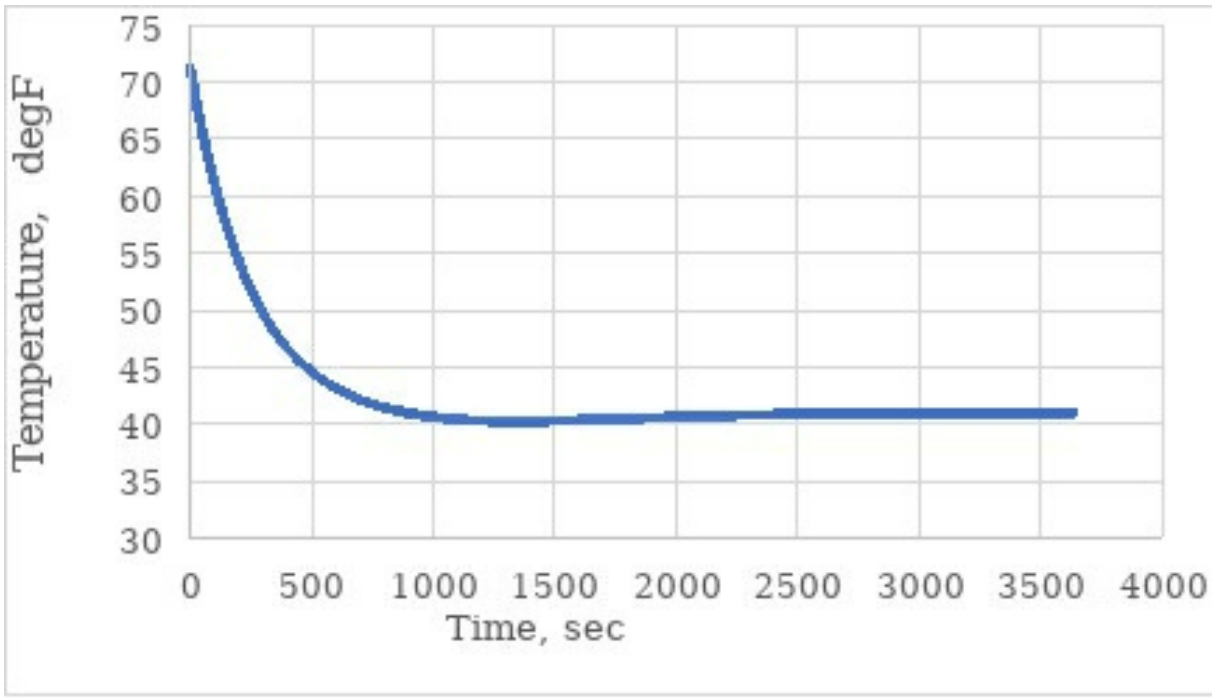


Figure 8.5 Transient temperature change in the insulating box after ice bags were added.

The air temperature has reached its low, equilibrium value after about 1000 seconds, which is 17 minutes. After this point, the air temperature will be as stable as it is going to be. It stabilized at about 41 degF.

I measured the ambient air temperature inside the box for another hour, long after equilibration with four sensors. The calibration terms were directly from the datasheet. Figure 8.6 shows the measured air temperature inside the box over a 1-hour period.

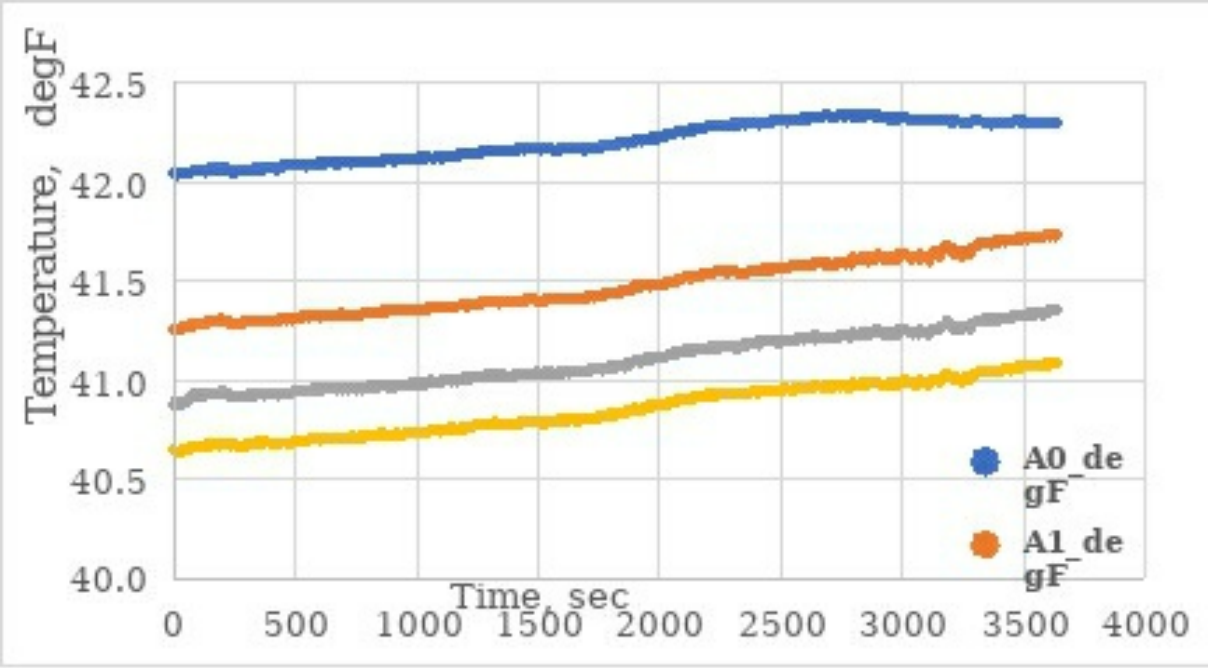


Figure 8.6 The measured temperature on four sensors using the default calibration in an ice bag insulated box. Scale is 0.2 degF per division.

With no special calibration, the temperature of all four sensors was within 0.7 degF of each other. The air drifted up in time by about 0.4 degF over the 1-hour period. This is a measure of what sort of stability we might expect to see using ice bags as a thermal buffer.

8.3 Temperature stability in ambient air

In the very first temperature stability measurements, we saw that the ambient air temperature in my lab varied by as much as 0.5 degF over an hour due to the furnace in the house cycling.

How stable would the temperature be if the furnace did not cycle on? Obviously, it would depend on the thermal environment of the room and the circulation of air currents.

In the next experiment, two sensors were left out exposed in the air and two sensors were placed between a folded towel to protect it from air currents and provide a more stable environment. The setup is shown in Figure 8.7

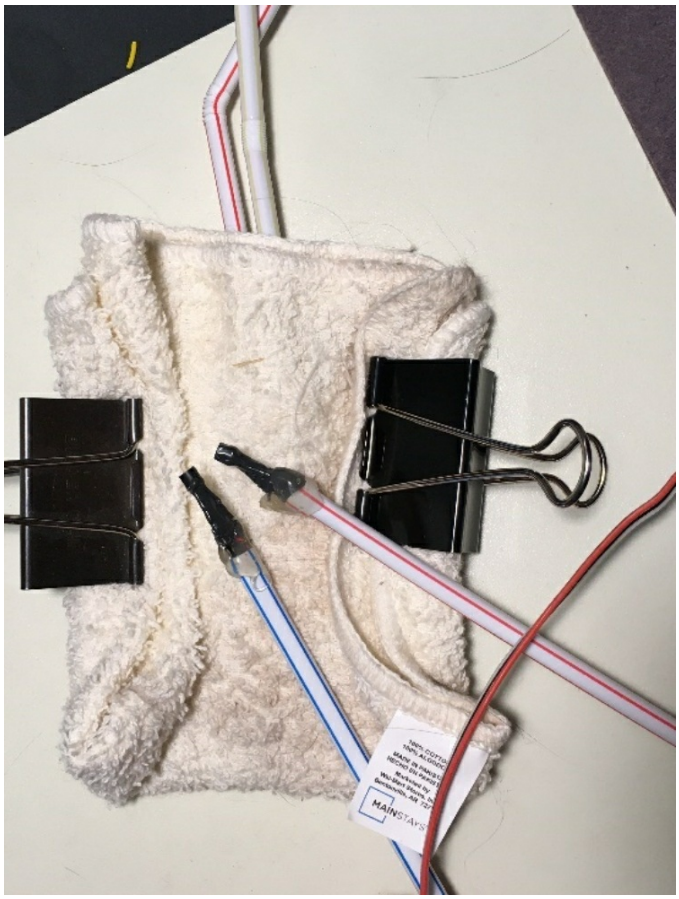


Figure 8.7. Experiment to compare the temperature stability of the ambient air and sensors inside a towel.

In these measurements, the datasheet default calibration terms were used each sensor. Since the sensors do not generate much heat, they are all at ambient temperature. However, the short-term fluctuations were much less with the sensors inside a towel. This is shown in Figure 8.8.

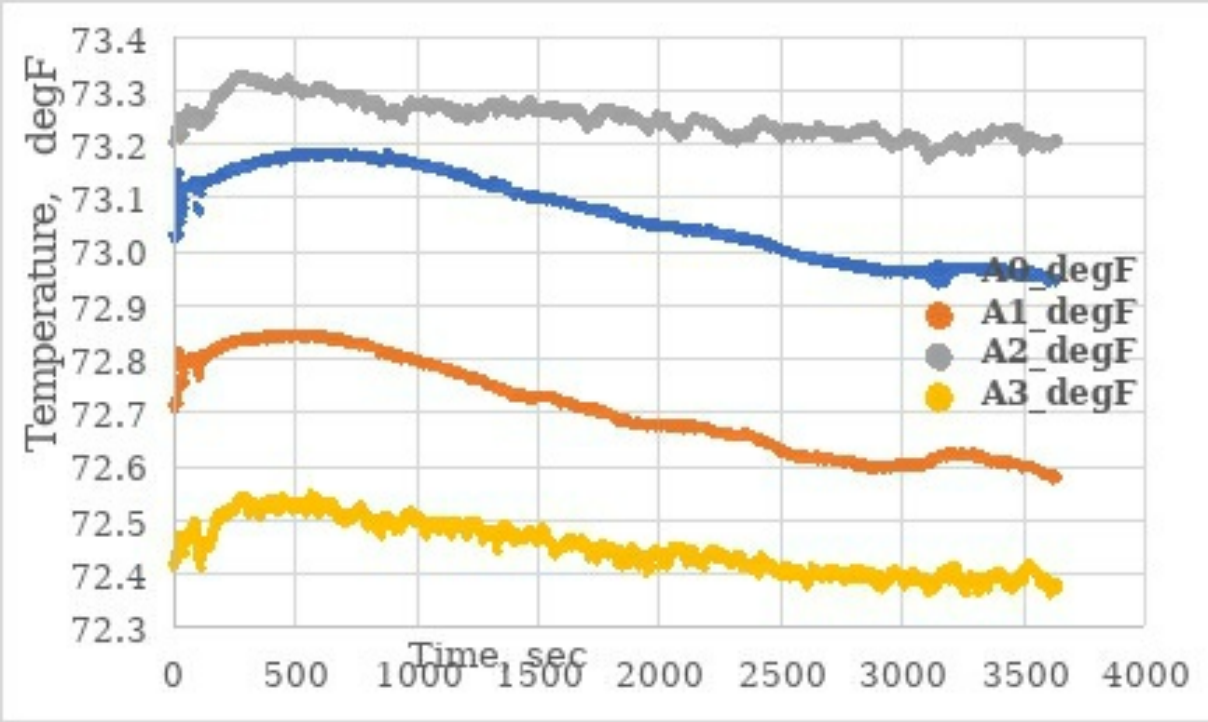


Figure 8.8. The measured temperature in the air of four sensors. The middle two are with the sensors inside a cloth while the other two sensors were exposed to ambient air.

Small air currents in the room generated small temperature fluctuations which are apparent in the sensors exposed to ambient air. These are absent in the two sensors wrapped inside the towel.

Overall, the ambient air temperature at this particular time varied by about 0.2 degF.

This is about the stability of the ambient air temperature measured when the sensors were inserted inside the insulating box, but with no ice bags and slowly circulating air. Figure 8.9 is a measure of the temperature inside the otherwise empty insulating box.

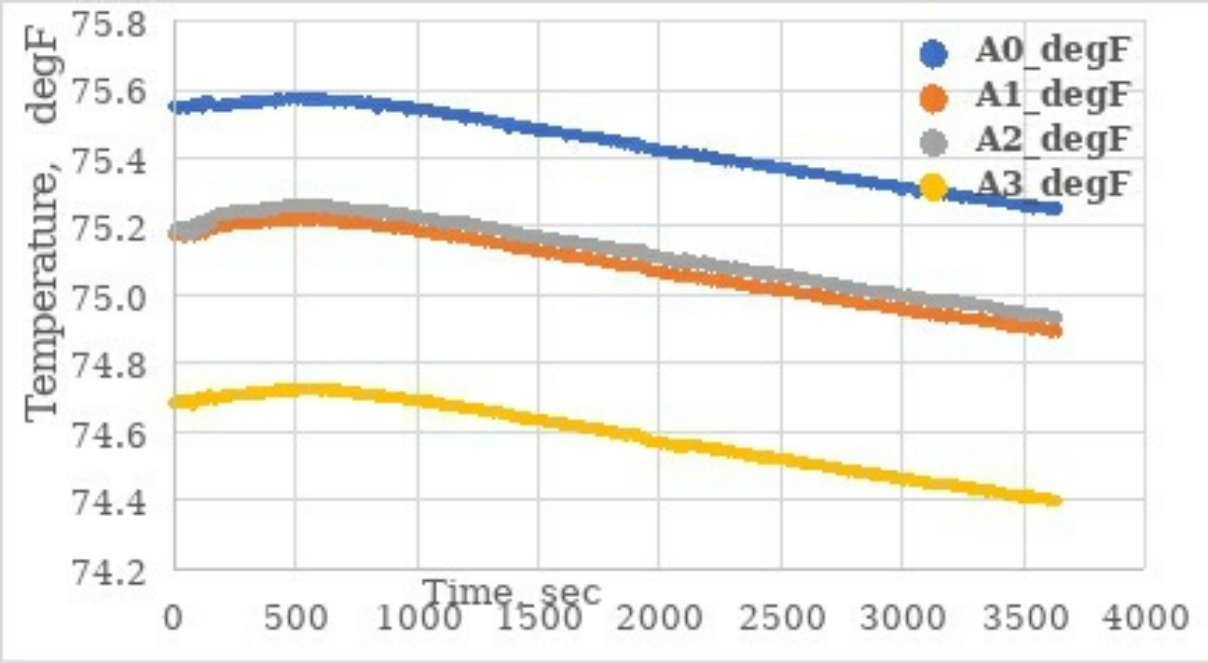


Figure 8.9 Stability of the air in an insulating box with slow fan air flow.

8.4 Cooling of coffee cups

In this last series of experiments, I wanted to evaluate how quickly different cups would cool after adding hot water.

I compared two different cups, an insulating cup and a ceramic coffee cup. The setup is shown in Figure 8.10.

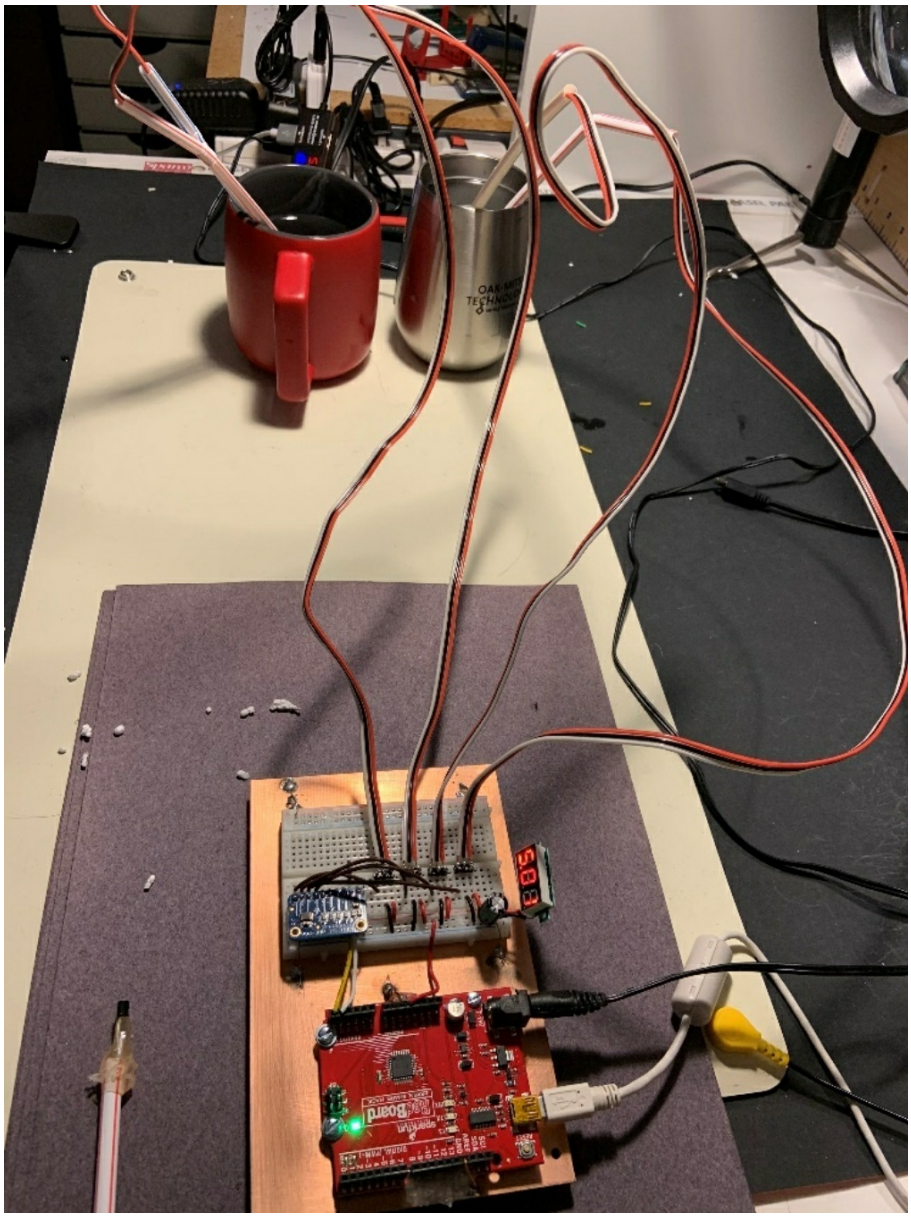


Figure 8.10 Set up to measure the cooling rate in a coffee mug and insulating cup.

Two sensors were added to each vessel. I used a single temperature calibration point for each sensor. I started with the same volume of hot water from the tap. By the time I carried the cups to the lab bench from the sink, the temperature of the water in the ceramic mug had already cooled due to the heat capacity of the ceramic mug.

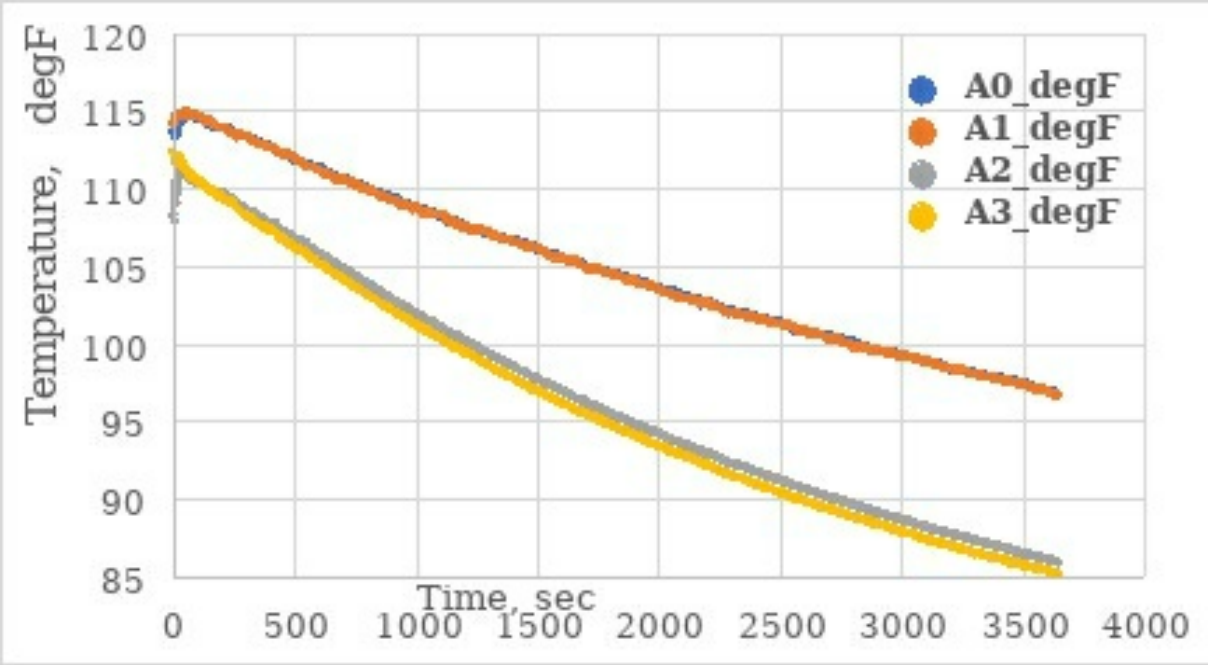


Figure 8.11 Cooling of a thermos and ceramic coffee cup measured by two probes in each cup. The ceramic mug is at the lower temperature.

From the first 1000 seconds, we can get a rough comparison of the cooling rates between these two cups.

The temperature change in the thermos cup is about (115-108 degF)/(16.7 min) = 0.42 degF/min drop.

In the same period, the temperature drop in the ceramic mug was (112-101 degF)/16.7 min = 0.66 degF/min.

These changes are only linear near the beginning when the temperature changes are large.

Chapter 9. Conclusions

In this issue of the HackingPhysics Journal, we've gone into details exercising the ADS1115 16-bit ADC with programmable gain amplifier.

For the astounding price if just $2, you can get started with high performance measurements down to the uV range, with an absolute accuracy to better than 1 mV that just ten years ago would have cost more than $2,000.

I purposefully used the lowest cost Arduino, the Uno, as the controller for this ADC to demonstrate what is possible with a low-cost system.

The rest of the hardware was not complicated, just using a solderless breadboard next to the Arduino Uno.

The measurements are directly imported and automatically plotted and saved in excel, ready for analysis.

While evaluating the voltage stability of a variety of sources, we found the AD584 chip, for $1, offered stability to 10 uV and absolute accuracy better than 1 mV.

If you need a stable, but adjustable voltage, using a resistor pot is an option, but just wait for its mechanical transients to settle. Afterall, a stability of 10 uV, out of 1 V for a 1 cm circumference resistor slider inside the pot, corresponds to a position accuracy of 0.1 um.

When we applied this precision voltage measurement system to the TMP36 temperature sensor, we found that after we reduced the rf pick up sensitivity to voltage measurements we can see changes of less than 0.05 degF.

The expected temperature stability in still air is on the order of a about 0.2 degF over an hour.

This combination of 16-bit resolution, x16 programmable gain amplitude and a sketch that averages the measurements over a few power line cycles is a powerful data acquisition system to measure voltage variations from sensors of any sort from seconds to hours.

All this for a cost of less than $10 and using the sketches included in this issue as a getting started place.

Here are some suggested experiments to explore with this very sensitive system:

1. *Measure the voltage stability of other voltage sources like batteries, diodes and regulators*
2. *How much variation is there in the voltage of a batch of new batteries?*
3. *Measure the electrochemical voltage eof different metals immersed in water with different levels of salt or acidity*
4. *Does hot water freeze faster than cold water? This is the Mpemba effect.*
5. *Does a watched bot boil slower than a pot not watched?*
6. *Does cold water boil faster than hot water?*
7. *Which coffee cup keeps coffee hotter longer?*
8. *Which will keep your coffee hotter longer, adding the cream right away after pouring the coffee or right before you are going to drink it?*
9. *How effective are thermos vessels at keeping hot water hot? What is the best you can do?*

Chapter 10. Your Next Steps

This brings us to the conclusion of this issue of the HackingPhysics Journal.

My hidden agenda, which I tried not to hide too well, is that Arduinos are incredibly powerful building block elements upon which to launch just about any sort of electronics project you want.

If you start out developing good habits, you will develop a style which accelerates you up the learning curve and enables you to tackle ever bigger and more complex projects.

Check out my other eBooks in this series: *Saturday Afternoon Low-Cost Electronics Projects*.

Arduinos Without Tears

Science Experiments with Arduinos Using a Multi-Function Board

Hacking Physics Journal Vol 1 No 1 (Jan 2020): Simple, low-cost and high-performance Arduino analog measurements

Hacking Physics Journal Vol 1 No 2 (April 2020): Arduino Plug and Play Experiments